US006032197A

United States Patent [19]

Birdwell et al.

[11] **Patent Number: 6,032,197**

[45] **Date of Patent: Feb. 29, 2000**

[54] **DATA PACKET HEADER COMPRESSION FOR UNIDIRECTIONAL TRANSMISSION**

[75] Inventors: **Kenneth J. Birdwell**; **Ruston Panabaker**, both of Bellevue; **Brian Moran**, Issaquah; **David Feinleib**, Redmond, all of Wash.

[73] Assignee: **Microsoft Corporation**, Redmond, Wash.

[21] Appl. No.: **08/937,476**

[22] Filed: **Sep. 25, 1997**

[51] **Int. Cl.**7 .... **G06F 13/00**; G06F 12/06

[52] **U.S. Cl.** **709/247**; 709/216; 709/217; 709/246

[58] **Field of Search** 709/246, 247, 709/218, 231, 216, 217, 219, 223, 232; 370/477; 707/10

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,864,572	9/1989	Rechen et al.	714/701
5,049,881	9/1991	Gibson et al.	341/95
5,293,379	3/1994	Carr	370/474
5,455,684	10/1995	Fujinami et al.	386/111
5,497,404	3/1996	Grover et al.	348/845.1
5,835,730	11/1998	Grossman et al.	709/247
5,938,736	8/1999	Muller et al.	709/243

OTHER PUBLICATIONS

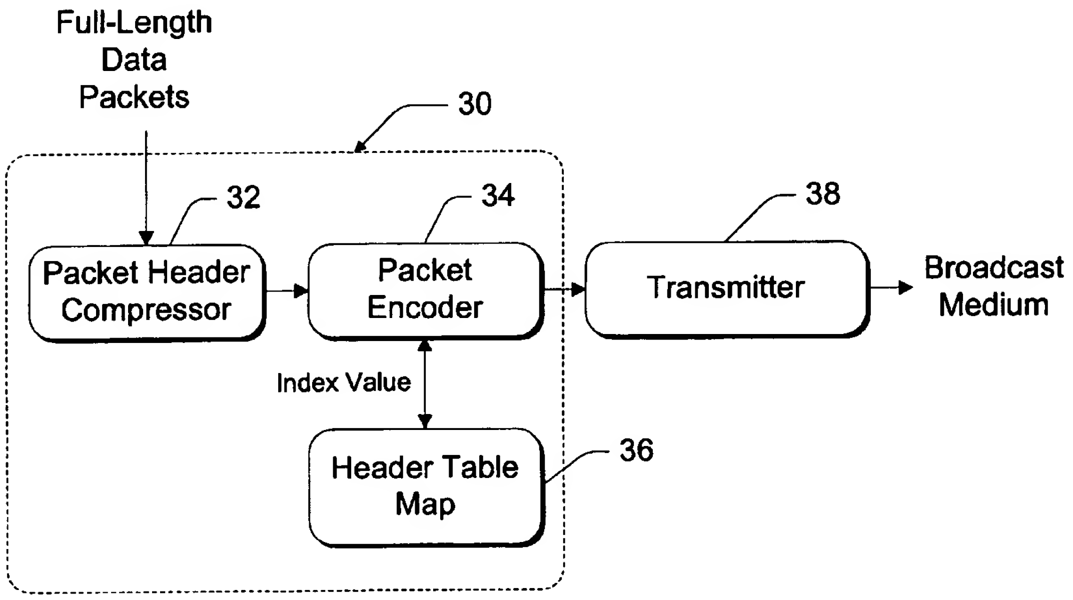
Jacobson V., "Compressing TCP/IP Headers for Low-Speed Serial Links", Feb. 1990.

Primary Examiner—Zarni Maung
Assistant Examiner—Jason D. Cardone
Attorney, Agent, or Firm—Lee & Hayes, PLLC

[57] **ABSTRACT**

A broadcast transmission system transmits data packets from a server to a client over a unidirectional broadcast network. The system transmits both full-length data packets, which have uncompressed headers, and reduced-length data packets, which have compressed headers derived from associated uncompressed headers. The server compresses the data packets by compressing the packet header. Compressed packet headers contain fewer header fields than their associated uncompressed headers. The server transmits a series of intermixed full-length and reduced-length packets to the client. As the packets are received, the client determines whether the packets are full-length or reduced-length. If the packet is full-length, the client stores the uncompressed header in a header table. If the packet is reduced-length, the client rebuilds the compressed header from its corresponding uncompressed headers in the header table.

42 Claims, 5 Drawing Sheets



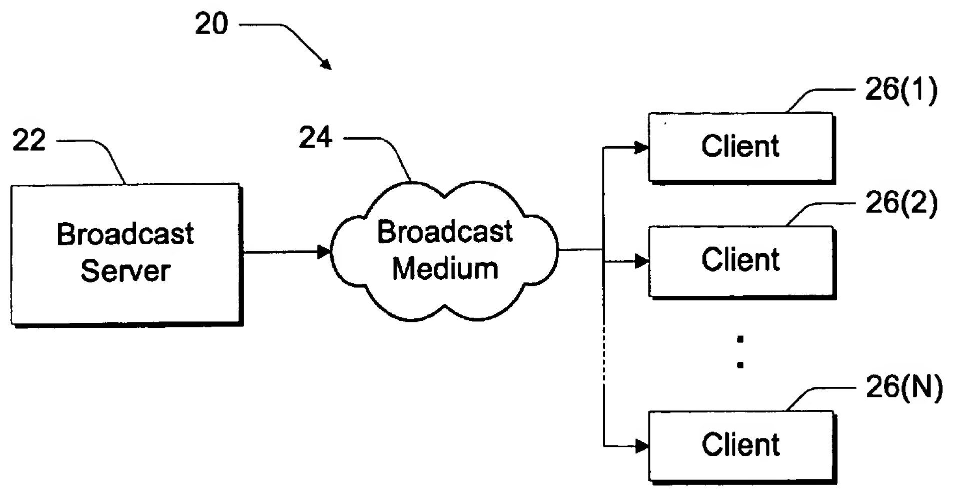

20
22
24
Broadcast Server
Broadcast Medium
26(1)
Client
26(2)
Client
26(N)
Client

Fig. 1

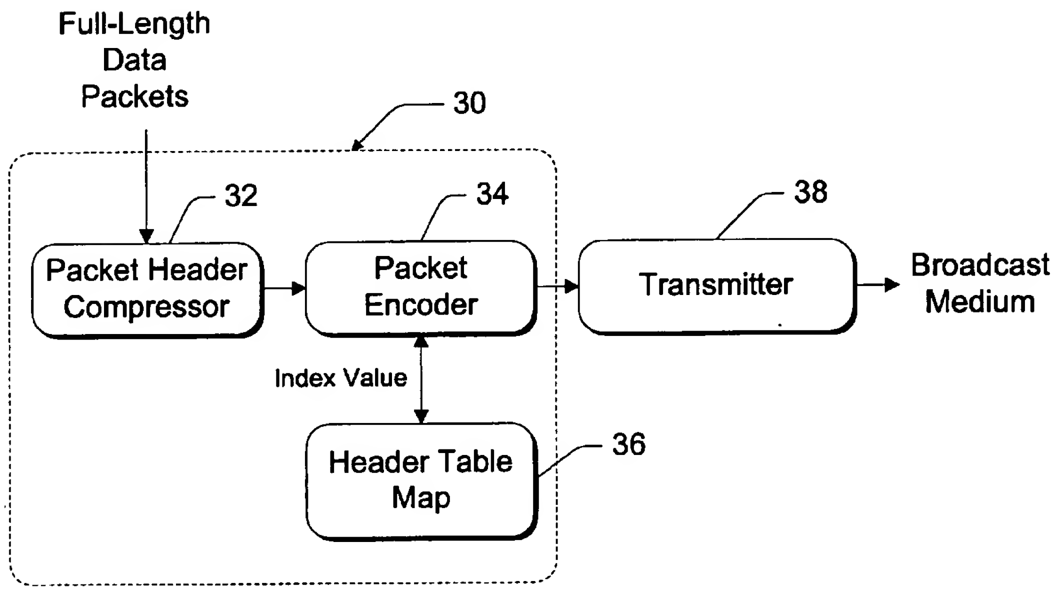

Full-Length Data Packets
30
32
34
38
Packet Header Compressor
Packet Encoder
Transmitter
Broadcast Medium
Index Value
Header Table Map
36

Fig. 2

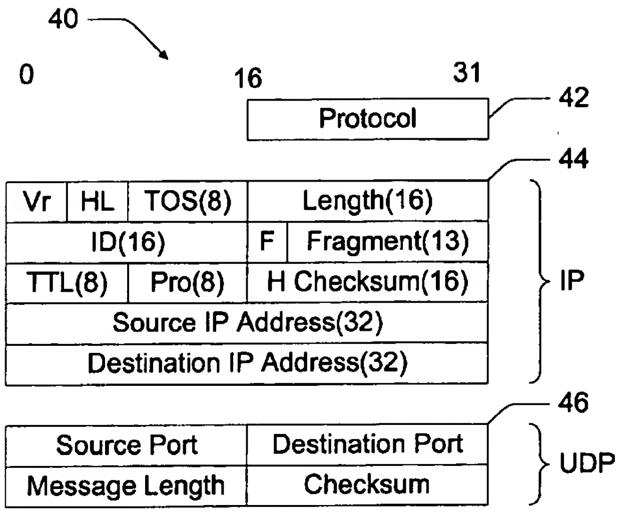


Fig. 3
Prior Art

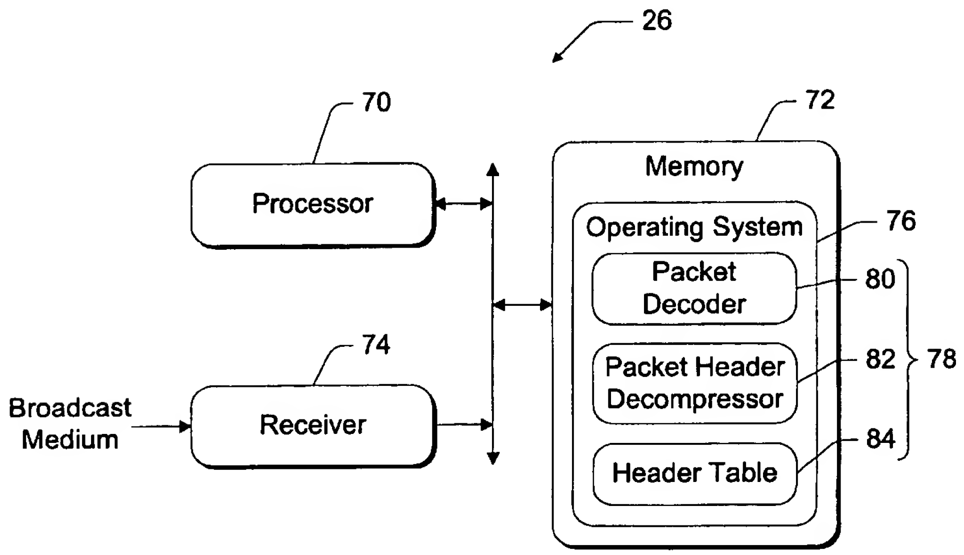


Fig. 6

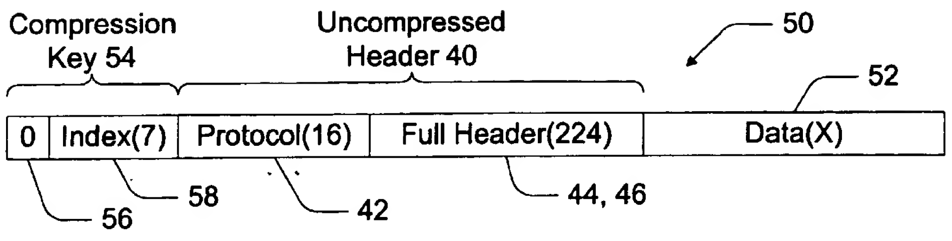

Compression Key 54
Uncompressed Header 40
50
52
0
Index(7)
Protocol(16)
Full Header(224)
Data(X)
58
56
42
44, 46

Fig. 4

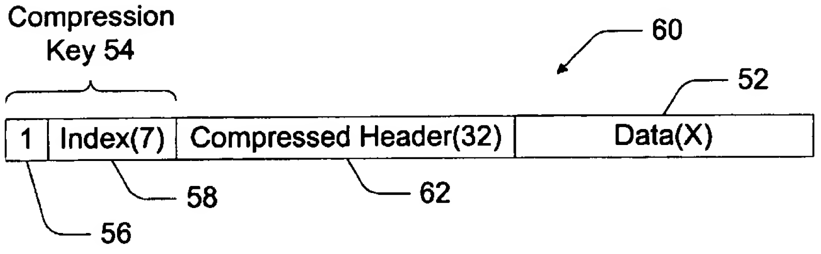

Compression Key 54
60
52
1
Index(7)
Compressed Header(32)
Data(X)
58
62
56

Fig. 5

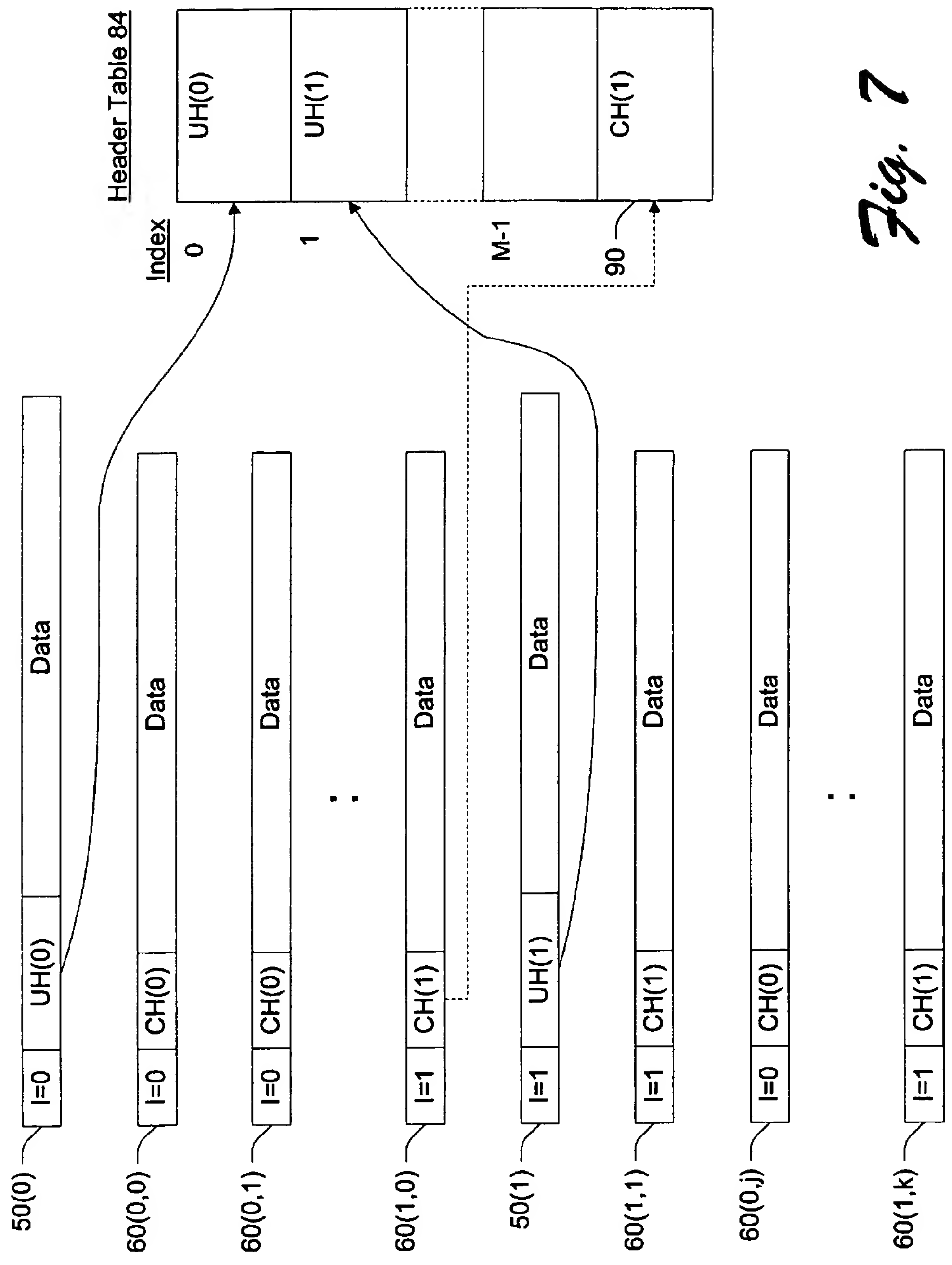

Header Table 84
Index
0
1
M-1
90
UH(0)
UH(1)
CH(1)
50(0)
60(0,0)
60(0,1)
60(1,0)
50(1)
60(1,1)
60(0,j)
60(1,k)
I=0
I=1
CH(0)
Data

Fig. 7

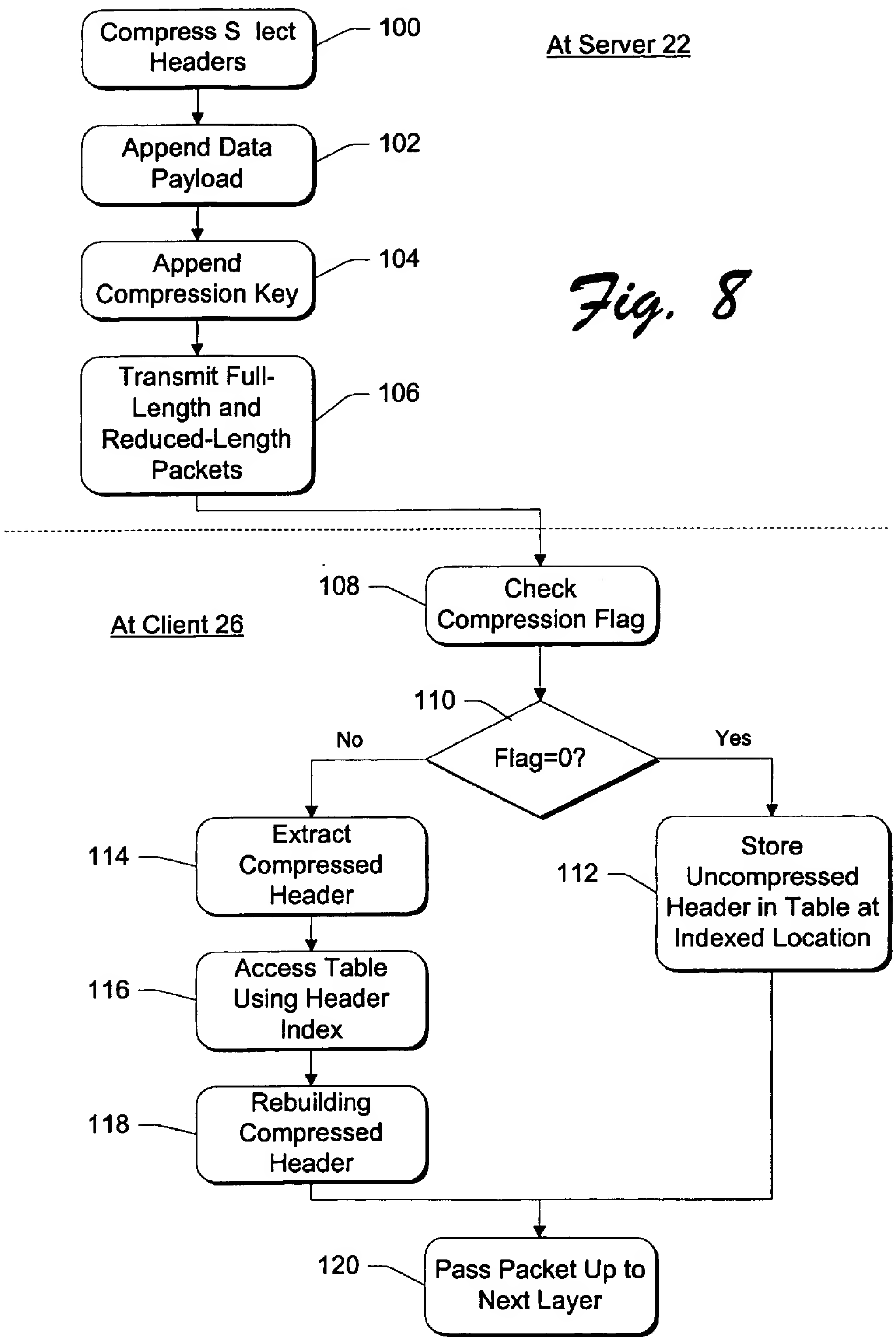

Compress S lect Headers
100
At Server 22
Append Data Payload
102
Append Compression Key
104
Transmit Full-Length and Reduced-Length Packets
106
108
Check Compression Flag
At Client 26
110
Flag=0?
No
Yes
114
Extract Compressed Header
112
Store Uncompressed Header in Table at Indexed Location
116
Access Table Using Header Index
118
Rebuilding Compressed Header
120
Pass Packet Up to Next Layer

Fig. 8

DATA PACKET HEADER COMPRESSION FOR UNIDIRECTIONAL TRANSMISSION

TECHNICAL FIELD

This invention relates to a broadcast transmission system in which data packets are sent over a broadcast medium to multiple clients. More particularly, this invention relates to a system and method for compressing headers in the data packets and delivering the compressed packets over the broadcast medium.

BACKGROUND OF THE INVENTION

Conventional computer networks are bi-directional, allowing data communication in both directions between servers and clients. Transmitting data over these bi-directional data networks has been a mainstay of computer technology for many years and the communication protocols are well established. Under conventional communication protocols, it is common for the client to initiate connection with the server and to request desired data from the server. As part of the request, the client sends information pertaining to how the data should be sent. For example, the client might include a client address, TCP port number, and so forth.

Digital data, whether transmitted over a wire-based distribution network (e.g., local area network, wide area network, cable, etc.) or a wireless distribution network (e.g., satellite, RF, paging, etc.), is typically packetized and sent over the network in individual packets. Some protocols call for fixed size packets, while other protocols utilize variable size packets. To improve transmission efficiency and to keep pace with the exploding demand for digital information, there is a constant design objective to pump increasingly more data through the same bandwidth pipeline over the network.

One way to achieve this objective is through packet compression. Packets are compressed at the server, transmitted in their compressed state over the network, and decompressed at the client. Apart from compressing whole packets, another solution is partial packet compression in which portions of the packet, such as a header or a data payload, are compressed. One technique for compressing packet headers is discussed in an article by V. Jacobson, entitled "Compressing TCP/IP Headers for Low-Speed Serial Links," and found at the web site http://ds.internic.net/rfc/rfc1144.txt. The Jacobson technique provides an elaborate and complex compression scheme that reduces a 40-byte TCP/IP (Transmission Control Protocol/Internet Protocol) packet header to a three-byte compressed header. The compressed header has an encoded change to the packet ID, a TCP checksum, a connection number, and a change mask. The hardware and/or software used to implement the Jacobson technique must perform sophisticated computations that compress the 40-byte header to the three-byte compressed header, and then subsequently decompress the compressed header to reproduce the uncompressed header.

Apart from the classic bi-directional data networks, there is an increasing interest in the use of broadcast or multicast networks to deliver computer data and other content to clients. These types of distribution networks are unidirectional. Data flows from the server to the clients, but no return communication is possible over the same communication path. As a result, a unidirectional broadcast network cannot support the common protocols used for two-way communication over a bi-directional network, such as client-driven

connections and data requests, because the clients are unable to communicate over the broadcast communication link to the server.

Like the bi-directional networks, there is benefit in sending compressed data packets over unidirectional broadcast networks. This invention is directed to a packet header compression technique that improves upon the Jacobson compression scheme, and that is specifically tailored for unidirectional broadcasts.

SUMMARY OF THE INVENTION

This invention concerns a broadcast transmission system for transmitting data packets from a server to a client over a unidirectional broadcast network. The system facilitates transmission of both full-length data packets, which have uncompressed headers, and reduced-length data packets, which have compressed headers derived from associated uncompressed headers.

In general, the server compresses the data packets by compressing the packet header. Compressed packet headers contain fewer header fields than the uncompressed headers. The server transmits a series of intermixed full-length and reduced-length packets over the broadcast medium to the clients. When a full-length packet is received, the client stores the uncompressed header in a header table. When a reduced-length packet is received, the client rebuilds the compressed header from the associated uncompressed header in the table from which the compressed header was originally derived. The uncompressed header is then appended to its data payload to effectively decompress the reduced-length data packet.

According to one implementation, the server has a packet header compressor to compress an uncompressed header into a compressed header. The uncompressed header has multiple fields. As an example, a UDP/IP (User Datagram Protocol/Internet Protocol) packet header has several well-defined IP fields—such as a version field, a header length field, a type of service field, a total length field, a packet identification field, a flag field, a fragment field, a time to live field, a protocol field, a header checksum field, a source IP address field, and a destination IP address field—and several well-defined UDP fields—such as a source port number field, a destination port number field, a UDP length field, and a UDP checksum field. Some of these fields do not change from packet to packet, rather only a subset of the fields changes.

The packet header compressor forms a compressed header from the fields of an associated uncompressed header. The compressed header contains one or more fields, which are subject to change from packet-to-packet, but not all of the fields in a normal uncompressed header. The fields that are common to both the compressed and uncompressed headers are otherwise identical. Compression is achieved by removing the non-changing header fields from the compressed header. For instance, in the case of compressing a UDP/IP header, the packet header compressor might form a compressed header having only the packet identification field, the flag field, and the fragment field, which change from packet to packet, while omitting the other IP and UDP fields.

The server also has a packet encoder to construct full-length and reduced-length data packets by appending data payloads and compression key blocks to the uncompressed and compressed headers. A full-length data packet includes a compression key block, an uncompressed header, and a data payload. A reduced-length data packet includes a compression key block, a compressed header, and a data payload. The compression key block has a compression bit value that identifies the packet as either a full-length data packet or a reduced-length data packet. The compression key block also contains a header index value that indexes to a memory location in a header table at a recipient where the uncompressed packet header is or will be stored. A transmitter resident at tle server transmits the full-length and reduced-length data packets over a unidirectional distribution medium to the clients.

Each client has a packet decoder to extract the compression key blocks from the data packets. If the compression bit value indicates that the packet is full-length, the packet decoder stores the uncompressed header in the memory location of the header table referenced by the corresponding header index value. On the other hand, if the compression bit value indicates that the packet is reduced-length, the client utilizes a packet header decompressor to reconstruct the uncompressed header. The packet header decompressor accesses the header table at a memory location indexed by the header index value contained in the reduced-length data packet. The memory location holds the uncompressed header from which the compressed header was derived. The packet header decompressor then reconstructs missing fields in the compressed header from the full set of fields in the associated uncompressed header. The full-length packets and decompressed packets are then passed onto the next layer in the protocol stack for further processing.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagrammatic illustration of a broadcast transmission system for broadcasting data packets from a server to multiple clients over a unidirectional broadcast medium.

FIG. 2 is a functional block diagram of a packet compressor located at the server to prepare data packets prior to transmission over the broadcast medium.

FIG. 3 is a diagrammatic illustration of a header constructed according to UDP/IP (User Datagram Protocol/Internet Protocol) format.

FIG. 4 is a diagrammatic illustration of a data structure for a full-length data packet type.

FIG. 5 is a diagrammatic illustration of a data structure for a reduced-length data packet type.

FIG. 6 is a block diagram of the client.

FIG. 7 is a diagrammatic illustration of a series of intermixed full-length and reduced length data packets and a header index table.

FIG. 8 is a flow diagram of a method for serving full-length and reduced-length data packets from a server to a client over a broadcast medium.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 shows a broadcast transmission system 20. Content is delivered in the form of data packets a broadcast server 22 over a broadcast medium 24 to multiple clients 26(1), 26(2), . . . , 26(N). Examples of transmittable content include computer data, audio, video, animation, bit maps or other graphics, applications or other executable code, text, hypermedia, or other multimedia types.

The broadcast medium 24 may comprise the entire distribution network between the server and client, or it may be a single link in a larger distribution network. Generally, the broadcast medium 24 is unidirectional in the sense that packets are delivered from the server 22 to the clients 26(1)–26(N) without requiring return communication from the clients. The broadcast medium 24 can be characterized as a shared, highly asymmetrical, network resource with a limited, if not completely absent, low speed return path that does not need to be active to receive broadcast transmissions.

The broadcast medium 24 can be implemented in a variety of ways. For instance, the broadcast network might be implemented as a wireless network configured for one-way transmission (i.e., satellite, radio, microwave, etc.). The broadcast network might also be a network that supports two-way communication (i.e., Internet, LAN (local area network), and WAN (wide area network)), but can be used for unidirectional multicasting from the broadcast server 22 to the clients 26(1)–26(N).

In general, the server 22 and the client 26 both have knowledge of the protocol and the block format of each packet. The server 22 compresses certain ones of the data packets by compressing their packet headers. Compressed packet headers contain fewer header fields than the uncompressed headers. The server 22 transmits both full-length data packet headers (which contain uncompressed headers) and reduced-length data packets (which contain compressed headers derived from the uncompressed headers) over the broadcast medium 24 to the clients. To realize sufficient benefit, a higher percentage of compressed, reduced-length packets are transmitted in comparison to full-length data packets. The client stores the uncompressed headers of the full-length data packets in a header table. The client then rebuilds compressed headers from the uncompressed headers in the table to decompress the reduced-length data packets.

FIG. 2 shows a packet compressor 30 implemented at the broadcast server 22. It includes a packet header compressor 32, a packet encoder 34, and a header table map 36. The packet compressor 30 can be implemented in hardware, software, or a combination of both. One example implementation is to incorporate the packet compressor into server software, such as Windows® NT from Microsoft Corporation, which executes on a server computer.

The packet header compressor 32 receives an incoming stream of full-length data packets that comprise a data payload and an uncompressed packet header. The data packets may be constructed according to one or multiple protocol formats. For discussion purposes, aspects of this invention are described in the context of a UDP/IP (User Datagram Protocol/Internet Protocol) format. However, this invention can accommodate other well-known formats as well, including TCP/IP, IPX/SPX, and NetBEUI. Moreover, one aspect of this invention concerns a header compression scheme that is capable of supporting multiple different formats individually or intermixed.

FIG. 3 shows an uncompressed header 40 constructed according to the UDP/IP format. A 16-bit protocol block 42 identifies the protocol format for the header 40. In this case, the protocol block 42 contains the protocol number "0x0800", which specifies the UDP/IP protocol. The header 40 also has a 160-bit IP portion 44, and a 64-bit UDP portion 46. The IP portion 44 consists of multiple IP fields: a 4-bit version field, a 4-bit header length field, an 8-bit type of service field, a 16-bit total length field, a 16-bit packet identification or sequence field, a 3-bit flag field, a 13-bit fragment field, an 8-bit time to live field, an 8-bit protocol field, a 16-bit header checksum field, a 32-bit source IP address field, and a 32-bit destination IP address field. The UDP portion 46 consists of multiple UDP fields: a 16-bit

identify protocol format

source port number field, a 16-bit destination port number field, a 16-bit UDP length field, and a 16-bit UDP checksum field.

The fields in the IP and UDP portions are well known, and are not discussed in detail. A detailed discussion of the UDP and IP formats is provided in a number of textbooks, such as Richard Stevens, *TCP/IP Illustrated, Volume* 1, Addison-Wesley Publishing Company, ©1994 and Douglas E. Comer, *TCP/IP, Volume* 1, *Principles, Protocols, and Architecture*, Third Edition, Prentice Hall, ©1995.

Some of the fields in the packet headers do not change from packet to packet, rather only a subset of the fields changes. In the case of the UDP/IP format, only the 16-bit packet identification field, the 3-bit flag field, and the 13-bit fragment offset field change from packet to packet. Other fields might change every few or more packets, but not every packet. In addition, some fields are redundant or non-essential and can be omitted or recomputed on the client side.

With reference again to FIG. 2, the packet header compressor 32 forms a compressed header from the fields of an associated uncompressed header. Preferably, the packet header compressor 32 forms a compressed header having those fields that are subject to change from packet-to-packet, but not all of the fields in a normal uncompressed header. In the UDP/IP example, the packet header compressor 32 forms a 32-bit compressed header containing only the 16-bit packet identification field, the 3-bit flag field, and the 13-bit fragment offset field. The other IP and UDP fields are omitted.

The fields that are common to both the compressed and uncompressed headers are identical. That is, the fields themselves are not compressed. The 16-bit packet identification field, for example, is the same in both uncompressed headers and compressed headers. Compression is achieved by removing the non-changing header fields from the compressed header. In the UDP/IP case, the packet header is compressed from 224 bits to 32 bits. By keeping the fields the same, the header compression technique simplifies packet header compression and decompression at the server and client.

In FIG. 2, the packet encoder 34 receives the data payloads and packet headers (compressed or uncompressed) from the packet header compressor 32 and forms full-length and reduced-length data packets that are ready for transmission over the broadcast medium. Full-length data packets have a data payload and an uncompressed header, whereas reduced-length data packets have a data payload and a compressed header. Reduced-length data packets may also be referred to in this disclosure as "compressed" data packets.

FIG. 4 shows a full-length data packet **50** that is constructed according to the UDP/IP format. The full-length data packet **50** includes an X-bit data payload **52** and an uncompressed UDP/IP header **40**. The uncompressed header **40** consists of the 224-bit IP and UDP headers **44** and **46**, and the 16-bit protocol block **42** (FIG. **3**).

FIG. **5** shows a compressed or reduced-length data packet **60** that is derived from the full-length data packet **50**. The reduced-length data packet **60** has the X-bit payload **52** and a compressed header **62**. In the UDP/IP case, the compressed header **62** is a 32-bit header comprising the 16-bit packet identification field, the 3-bit flag field, and the 13-bit fragment offset field.

The packet encoder **34** (FIG. **2**) appends a compression key **54** to each packet, regardless of whether the packet is

full-length or reduced length. As shown in FIGS. **4** and **5**, the compression key **54** has a compression bit value **56** and a header index value **58**. The compression bit value **56** identifies the packet as either a full-length data packet or a reduced-length data packet. In this example, the compression bit value is a one-bit compression flag that is a first binary value, such as a "0", when the data packet is full-length and a second binary value, such as a "1", when the data packet is reduced-length.

The header index value **58** references a memory location at the destination client. More particularly, the header index value **58** is used to reference an entry in a header table at the destination client. If the header index value **58** belongs to the full-length data packet **50**, the header index value designates an entry in the header table for storing the uncompressed header **40**. On the other hand, if the header index value **58** belongs to the reduced-length data packet **60**, the header index value designates an entry in the header table that stores (or will store) the associated uncompressed header **40** from which the compressed header **62** is derived.

With continuing reference to FIG. **2**, the packet encoder **34** derives one or more reduced-length data packets from a full-length data packet. A full-length data packet and a reduced-length data packet are said to be "associated", "corresponding" or other similar language if the compressed header in the reduced-length data packet is derived from the uncompressed header in the full-length data packet. Likewise, an uncompressed header and a compressed header are said to be "associated", "corresponding" or other similar language if the compressed header contains a subset of the fields found in the uncompressed header.

The packet compressor **30** includes a header table map **36**, which tracks or mirrors the entries in the header table at the client. When the packet encoder **34** assigns a header index value to a full-length data packet, the packet encoder **34** requests from the header table map **36** an index value to a new location in the header table for storing the uncompressed header. When the packet encoder assigns a header index value to a reduced-length data packet, the packet encoder **34** receives from the header table map **36** an index value to a location in the header table that stores the associated uncompressed header. Through the use of the header table map, the server is in complete control of when an uncompressed header is sent, and where it is stored at the client.

It is noted that the packet encoder **34** may append other headers and trailers to the data packets. For instance the packet encoder might add a trailer containing a CRC (cyclic redundancy check) value. The packet encoder prepares the packets for transmission over the broadcast medium.

The packet compressor **30** outputs both the full-length and reduced-length data packets to a transmitter **38** for transmission over the broadcast medium. The transmitter **38** may be implemented in many different ways, depending upon the network implementation. For instance, the transmitter **38** may be a satellite uplink, an RF transmitter, a microwave transmitter, a modem, a network card, and so forth.

FIG. **6** shows an exemplary configuration of a client **26** implemented as a broadcast-enabled computer. It includes a central processing unit having a processor **70** (e.g., x86 or Pentium®-family microprocessor from Intel Corporation) and memory **72**. The memory **72** generally includes volatile memory (e.g., RAM) and non-volatile program memory (e.g., ROM, disk drive, floppy disk drive, CD-ROM, etc.). The client **26** typically has one or more input devices (e.g., keyboard, mouse, etc.) and a computer display (e.g., VGA, SVGA), which are not shown in this figure. The client **26** also includes a digital broadcast receiver **74** (e.g., satellite dish receiver, RF receiver, microwave receiver, NTSC-VBI receiver, digital cable TV decoder, modem, network card, etc.) to receive the full-length and reduced-length data packets from the distribution medium.

The client **26** runs an operating system **76** that supports multiple applications. The operating system **76** is preferably a multitasking operating system that allows simultaneous execution of multiple applications. One preferred operating system is a Windows® brand operating system sold by Microsoft Corporation, such as Windows® 95 or Windows® NT or other derivative versions of Windows®. It is noted, however, that other operating systems may be employed.

In the FIG. **6** implementation, the operating system **76** incorporates a packet decompressor **78** to decompress data packets received by the receiver **74** from the broadcast medium. The packet decompressor **78** has a packet decoder **80**, a packet header decompressor **82**, and a header table **84**. Although the packet decompressor **78** is shown implemented in software as part of the operating system **76**, it may alternatively be implemented as a separate component in hardware and/or software.

The packet decoder **80** extracts the compression key **54** from each incoming data packet. If the compression flag **56** in the compression key **54** indicates that the packet is full-length, the packet decoder stores the appended uncompressed header **40** in the header table **84** at a memory location referenced by the header index value **58**. If the compression flag **56** in the compression key **54** indicates that the packet is reduced-length, the packet decoder passes the appended compressed header **62** and header index value **58** to the packet header decompressor **82**.

It is noted that the packet decoder **80** may perform other functions, such as error analysis, stripping away carrier headers that are not needed by the upper layer protocols, and the like.

The packet header decompressor **82** reconstructs an uncompressed header from the compressed header received from the packet decoder **80**. The packet header decompressor **82** utilizes the header index value **58** to reference the memory location in the header table **84** that stores the associated uncompressed header. The decompressor **82** rebuilds the header by adding the missing fields in the compressed headers from the full set of fields in the associated uncompressed headers store in the table. In the UDP/IP case, the packet header decompressor **82** adds all of the fields in the IP portion **44** and UDP portion **46**, excepting the packet identification field, the flag field, and the fragment/offset field from the IP portion **44** that are already present in the uncompressed header.

FIG. **7** illustrates the packet decompressing process implemented at the client. A series of intermixed full-length packets **50** and reduced-length packets **60** is received at the client from the broadcast medium. The first full-length packet **50**(0) has an index value of "0" (i.e., the "I=0" block) which references entry 0 in the header table **84**. The packet decoder **80** stores the uncompressed header UH(0) from data packet **50**(0) at entry 0 in the header table **84**, as indicated by the solid arrow.

The next packet in the series is a reduced-length packet **60**(0,0). The notation "**60**(x,y)" means the y^{th} reduced-size packet **60** that is derived from the x^{th} full-length data packet. The reduced-length packet **60**(0,0) carries an index value of "0" (i.e., The "I=0" block) because the compressed header CH(0) is associated with the uncompressed header UH(0) that is stored in table entry 0. Accordingly, the packet header decompressor uses the index value of 0 to access the header table **84** and locate the associated uncompressed header UH(0) for rebuilding the compressed header UH(0). This process is continued for the next reduced-length packet **60**(0,1), and so on.

The next full-length packet **50**(1) has an index value of "1" (i.e., the "I=1" block) which references entry 1 in the header table **84**. The packet decoder **80** stores the uncompressed header UH(1) from data packet **50**(1) at entry 1 in the header table **84**. Any reduced-length data packet **60**(1,y) that is derived from full-length packet **50**(1) also carries a header index value of 1 to locate the associated uncompressed header UH(1).

The header table **84** has additional space **90** to temporarily cache compressed headers in the event that their associated uncompressed headers are not yet received at the client and stored in the table. If a compressed header arrives at the client before the uncompressed header from which it is derived, the compressed header is cached until the associated uncompressed header is indexed into the table. In FIG. **7**, the reduced-length data packet **60**(1,0) is received prior to its corresponding full-length data packet **50**(1). The packet decoder **80** places the compressed header CH(1) in a compressed header cache **90**. After the associated uncompressed header UH(1) is received and stored in entry 1, the packet decompressor **82** retrieves the cached compressed header CH(1) from cache **90** and rebuilds it from the uncompressed header UH(1).

The header table **84** has a finite number of entries "M". The uncompressed headers UH(0), UH(1), . . . UH(M−1) stored in the table can be configured to time out or expire after a preset duration. This ensures that only up-to-date uncompressed headers are maintained in the table. Since the server is in complete control of when an uncompressed header is sent, and where it is stored in the table, the server can send new uncompressed header entries as desired. Additionally, the table **84** can be implemented to help manage the entries. For instance, the header table **84** may manage entries according to a first-in-first-out (FIFO) protocol, so that the newest header overwrites the oldest header. However, other protocols may be used, such as one based on frequency of use whereby newly added headers overwrite the least frequently used headers.

One advantage of this compression scheme is that it is not protocol dependent. The UDP/IP format is used herein as an example, but many other formats may be used. Moreover, the compression scheme can accommodate simultaneous transmission of different protocols, so long as the client and server are both aware of the different types. In FIG. **7**, for example, the first full-length data packet **50**(0), and its reduced-length derivatives **60**(0,y), may be encoded using one format and the second full-length data packet **50**(1), and its reduced-length derivatives **60**(1,y), may employ a different format.

FIG. **8** shows exemplary steps in a method for serving full-length and reduced-length data packets from a server to a client over a broadcast medium. At step **100**, the server **22** compresses select packet headers. The server **22** can employ many different techniques for choosing which packet headers to compress. One approach is to identify groups of packets in which only a predetermined subset of fields change and for each group, leave one uncompressed header and compress the remaining headers from the uncompressed header. Another technique is to try to compress P headers for every uncompressed header.

At steps **102** and **104**, the server **22** appends the data payload and compression key to the uncompressed or compressed header to form full-length or reduced-length data packets, respectively. The server **22** transmits a series of intermixed full-length and reduced-length data packets over the broadcast medium **24** to the clients **26(1)–26(N)** (step **106** in FIG. **8**).

At step **108** in FIG. **8**, each client **26** receives the packets and checks the compression flag in the compression key. If the flag is a binary "0" indicating that the packet is full-length (i.e., the "yes" branch from step **110**), the client **26** extracts the uncompressed header and stores it in the header table **84** at an entry indexed by the header index value (step **112**). Conversely, if the flag is a binary "1" indicating that the packet is reduced-length (i.e., the "no" branch from step **110**), the client **26** extracts the compressed header from the data packet (step **114**) and uses the header index value to access the associated uncompressed header in the header table **84** (step **116** in FIG. **8**). The client **26** uses the uncompressed header to reconstruct the compressed header (step **118**). As the reduced-length packets are decompressed, the client passes all packets, including the decompressed and full-length packets, up to the next protocol layer in the protocol stack (step **120** in FIG. **8**).

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

We claim:

1. A method for serving full-length and reduced-length data packets from a server to a client over a unidirectional broadcast medium, each full-length data packet having a data payload and an uncompressed header with multiple fields, the method comprising the following steps:

at the server:

compressing an uncompressed header to form an associated compressed header having a subset of one or more of the fields so that the fields that are common to the associated uncompressed and compressed headers are identical;

forming a reduced-length data packet having the compressed header and a data payload;

appending a compression key to each of the full-length and reduced-length data packets, the compression key specifying whether the appended data packet is full-length or reduced-length and whether the full-length and reduced-length data packets have associated uncompressed and compressed headers;

transmitting the full-length and reduced-length data packets from the server to the client over a unidirectional broadcast medium;

at the client:

extracting the uncompressed header from the full-length data packet;

storing the uncompressed header;

extracting the compressed header from the reduced-length data packet; and

decompressing the compressed headers by adding fields from the uncompressed header stored at the client that is indicated by the compression keys as being associated to the compressed header, the added fields restoring the subset of fields contained in the compressed header to said multiple fields contained in the uncompressed header.

2. A method as recited in claim **1**, wherein the appending step comprises the step of appending a two-block compression key having a first block that specifies whether the appended data packet is full-length or reduced-length and a second block containing an index to a table at the client, whereby the associated uncompressed and compressed headers are indexed to a same location in the table.

3. A method as recited in claim **2**, wherein the storing step comprises the step of storing the uncompressed header in the table, and further comprising the step of subsequently removing the uncompressed header from the table to make room for new uncompressed headers.

4. A method as recited in claim **1**, further comprising the step of temporarily caching the compressed header in an event that the uncompressed header associated with the cached compressed header has not yet been received at the client.

5. A method as recited in claim **1**, further comprising the step of transmitting a higher percentage of the reduced-length data packets in comparison to the full-length data packets.

6. A method as recited in claim **1**, further comprising the step of forming and transmitting multiple groups of full-length and reduced-length data packets, wherein each said group of data packets is formatted according to a different transmission protocol.

7. Computer-readable media located at the server and the client having computer-executable instructions for performing the steps of the method as recited in claim **1**.

8. A reduced-length data packet having a compressed header embodied on a transmission medium and constructed according to the steps performed at the server in the method as recited in claim **1**.

9. A method for compressing a data packet for transmission over a broadcast medium, the data packet comprising a data payload and an uncompressed packet header having multiple fields, the method comprising the following steps:

forming a compressed packet header comprising at least one, but not all, of the fields from the uncompressed packet header;

appending the data payload to the compressed packet header; and

appending a compression key to the compressed packet header, the compression key having a first block containing information to inform a receiver that the appended packet header is a compressed packet header and a second block containing an index to a table at the receiver that identifies the memory location of the uncompressed packet header.

10. A method as recited in claim **9**, further comprising the step of transmitting as a reduced-length data packet comprising the compressed packet header, the data payload, and the compression key.

11. A reduced-length data packet embodied on a transmission medium and constructed according to the steps performed at the server in the method as recited in claim **10**.

12. A compressed data packet embodied on a computer-readable medium constructed according to the steps performed at the server in the method as recited in claim **9**.

13. A computer programmed to perform the steps of the method as recited in claim **9**.

14. A computer-readable medium having computer-executable instructions for performing the steps of the method as recited in claim **9**.

15. A method for compressing an uncompressed packet header of a UDP/IP data packet to be transmitted over a broadcast medium, the uncompressed packet header having an IP portion containing multiple IP fields and a UDP portion containing multiple UDP fields, the method comprising the following steps:

forming a compressed packet header comprising at least one, but not all, of the IP fields in the uncompressed packet header; and

appending a compression key to the compressed packet header, the compression key comprising a first block containing information to inform a receiver that the appended packet header is a compressed packet header and a second block containing an index to a table at the receiver that identifies where the uncompressed packet header is or will be stored.

16. A method as recited in claim **15**, wherein the IP fields include a packet identification field and a fragment field, the forming step comprises the step of forming a compressed packet header consisting of the packet identification field and the fragment field.

17. A method as recited in claim **15**, wherein the IP fields include a packet identification field and a fragment field, further comprising the following steps:

forming a 32-bit compressed packet header consisting of the packet identification field and the fragment field;

appending a 7-bit index value to the compressed packet header, the index value containing an index to a table at a receiver that identifies where the uncompressed packet header is or will be stored; and

appending a one-bit compression flag to the index value that is used to inform the receiver that the appended packet header is a compressed packet header.

18. A method as recited in claim **15**, further comprising the step of appending a data payload to the compressed packet header.

19. A method as recited in claim **18**, further comprising the step of transmitting as a reduced-length data packet comprising the compressed packet header, the data payload, and the compression key.

20. A reduced-length data packet embodied on a transmission medium and constructed according to the steps performed at the server in the method as recited in claim **19**.

21. A compressed data packet embodied on a computer-readable medium constructed according to the steps performed at the server in the method as recited in claim **15**.

22. A computer programmed to perform the steps of the method as recited in claim **15**.

23. A computer-readable medium having computer-executable instructions for performing the steps of the method as recited in claim **15**.

24. A broadcast transmission system for transmitting full-length and reduced-length data packets from a server to a client over a broadcast medium, comprising:

a packet header compressor resident at the server to compress an uncompressed header with multiple fields into an associated compressed header with at least one, but not all, of the fields from an associated uncompressed header;

a packet encoder resident at the server to form full-length and reduced-length data packets, the full-length data packets including a data payload and an uncompressed header and the reduced-length data packets including a data payload and a compressed header, the packet encoder further including with each full-length data packet a header index value, the packet encoder further including with each reduced-length data packet the header index value of the full-length data packet that contains the uncompressed header that is associated with the compressed header within the reduced-length data packet;

wherein the packet encoder appends a first bit value to the full-length data packets and a second bit value to the reduced-length data packets to differentiate the full-length and reduced-length data packets;

a transmitter at the server to transmit the full-length and reduced-length data packets over the broadcast medium;

a receiver at the client to receive the full-length and reduced-length data packets from the distribution medium; and

a packet decoder at the client to extract the header index values from the full-length data packets and store the uncompressed headers of the full-length data packets in memory locations referenced by corresponding header index values; and

a packet header decompressor at the client to reconstruct uncompressed headers from the compressed headers in the reduced-length data packets, the packet header decompressor utilizing the header index values from the reduced-length data packets to reference the memory locations containing the associated uncompressed headers and reconstruct missing fields in the compressed headers from the fields in the associated uncompressed headers.

25. A broadcast transmission system as recited in claim **24**, wherein the packet decoder stores the uncompressed headers in a table at an entry identified by the corresponding header index values.

26. A broadcast transmission system as recited in claim **24**, wherein the packet decoder stores the uncompressed headers in a table according to a first-in-first-out protocol.

27. A broadcast transmission system as recited in claim **24**, wherein the packet decoder stores the uncompressed headers in a table, the stored uncompressed headers being considered expired after a predetermined time period elapses.

28. A broadcast transmission system as recited in claim **24**, wherein the packet header decompressor temporarily caches a compressed header in an event that the uncompressed header associated with the cached compressed header has not yet been received at the client.

29. A broadcast transmission system as recited in claim **24**, wherein the transmitter transmits a higher percentage of the reduced-length data packets in comparison to the full-length data packets.

30. A broadcast transmission system as recited in claim **24**, wherein groups of the full-length and reduced-length data packets are formatted according to different transmission protocols.

31. A broadcast transmission system as recited in claim **24**, wherein the data packets are formatted according to an UDP/IP protocol, the uncompressed headers have an IP portion containing multiple IP fields and a UDP portion containing multiple UDP fields and the compressed headers have at least one, but not all, of the IP fields in the associated uncompressed header.

32. A broadcast transmission system as recited in claim **31**, wherein the IP fields include a packet identification field and a fragment field and the compressed headers comprises the packet identification field and the fragment field.

33. A data packet compressor for compressing a data packet for transmission over a broadcast medium, the data packet comprising a data payload and an uncompressed packet header having multiple fields, comprising:

a packet header compressor to form a compressed packet header having at least one, but not all, of the fields from the uncompressed packet header;

a packet encoder to append a header index value to the compressed packet header, the header index value identifying for a recipient the uncompressed packet header from which the compressed packet header is formed; and

wherein the packet encoder appends a bit value to the header index value to indicate that the appended header is compressed;

the packet encoder further appending a data payload to the compressed packet header.

34. A data packet compressor as recited in claim **33**, wherein the data packet is formatted according to an UDP/IP protocol, the uncompressed header has an IP portion containing multiple IP fields and a UDP portion containing multiple UDP fields and the compressed headers has at least one, but not all, of the IP fields in the associated uncompressed header.

35. A data packet compressor as recited in claim **34**, wherein the IP fields include a packet identification field and a fragment field and the compressed header comprises the packet identification field and the fragment field.

36. A data packet compressor as recited in claim **34**, wherein the IP fields include a packet identification field and a fragment field, further comprising:

the packet header compressor forms a 32-bit compressed packet header consisting of the packet identification field and the fragment field; and

the packet encoder appends a 7-bit index value to the compressed packet header and a one-bit compression flag to inform the recipient that the appended packet header is compressed.

37. A software program embodied on a computer-readable medium having code which implements the data packet compressor as recited in claim **33**.

38. A data packet decompressor for decompressing a compressed data packet received from a broadcast medium, the compressed data packet comprising a data payload, a compressed packet header with one or more fields, a header index value, and a bit value to the header index value to indicate that the appended header is compressed, comprising:

a packet decoder to extract the compressed header from the compressed data packet;

a packet header decompressor to access a memory location referenced by the header index value, the memory location holding a full uncompressed packet header having multiple fields that include the fields in the compressed data packet and other fields missing from the compressed data packet; and

the packet header decompressor rebuilding a decompressed packet header from the compressed packet header by adding the missing fields from the memory location to the fields in the compressed data packet.

39. An operating system embodied on a computer-readable medium having code which implements the data packet decompressor as recited in claim **38**.

40. A network packet structure embodied on a computer-readable medium, comprising:

a full-length data packet type comprising:

a data payload;

an uncompressed header appended to the data payload, the uncompressed header having multiple fields;

a header index value to reference a memory location at a destination to hold the uncompressed header;

a first bit value to identify the packet as the full-length data packet type; and,

a reduced-length data packet type derived from the full-length data packet type comprising:

a data payload;

a compressed header appended to the data payload, the compressed header having at least one, but not all, of the fields found in the uncompressed header;

the header index value; and

a second bit value to identify the packet as the reduced-length data packet type.

41. A network packet structure as recited in claim **40**, wherein:

the uncompressed header is a UDP/IP header having an IP portion containing multiple IP fields and a UDP portion containing multiple UDP fields; and

the compressed header has at least one, but not all, of the IP fields in the associated uncompressed header.

42. A network packet structure as recited in claim **40**, wherein:

the uncompressed header is a UDP/IP header that includes a packet identification field and a fragment field; and

the compressed header consists of the packet identification field and the fragment field.

* * * * *

US006434168B1

(12) **United States Patent**
Kari

(10) **Patent No.: US 6,434,168 B1**
(45) **Date of Patent: Aug. 13, 2002**

(54) **DATA COMPRESSION ON A DATA CONNECTION**

(75) Inventor: **Hannu Kari**, Veikkola (FI)

(73) Assignee: **Nokia Telecommunications Oy**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/202,203**

(22) PCT Filed: **Jun. 3, 1997**

(86) PCT No.: **PCT/FI97/00345**

§ 371 (c)(1),
(2), (4) Date: **Dec. 7, 1998**

(87) PCT Pub. No.: **WO97/48212**

PCT Pub. Date: **Dec. 18, 1997**

(30) **Foreign Application Priority Data**

Jun. 7, 1996 (FI) 962381

(51) **Int. Cl.**^{7} **H04J 3/00**
(52) **U.S. Cl.** **370/521**; 348/568
(58) **Field of Search** 370/521; 379/93.08; 710/68; 709/247; 348/439.1, 14.13, 568

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,396,228 A		3/1995	Garahi	
5,452,287 A		9/1995	DiCecco et al.	
5,867,114 A	*	2/1999	Barbir	341/107
5,884,269 A	*	3/1999	Cellier et al.	704/501
5,933,104 A	*	8/1999	Kimura	341/87
5,956,504 A	*	9/1999	Jagadish et al.	707/101
5,974,179 A	*	10/1999	Caklovic	382/232
5,974,471 A	*	10/1999	Belt	710/1
6,002,719 A	*	12/1999	Parvulescu et al.	375/240
6,151,627 A	*	11/2000	McBride et al.	709/224
6,175,386 B1	*	1/2001	Van De Schaar-Mitrea et al.	348/563

FOREIGN PATENT DOCUMENTS

EP	595 406	5/1994
WO	94/14273	6/1994
WO	95/02873	1/1995

OTHER PUBLICATIONS

V. Jacobson, Compressing TCP/IP Headers for Low-Speed Serial Links (Request for Comments: 1144) (Feb. 1990); pp 1–25.

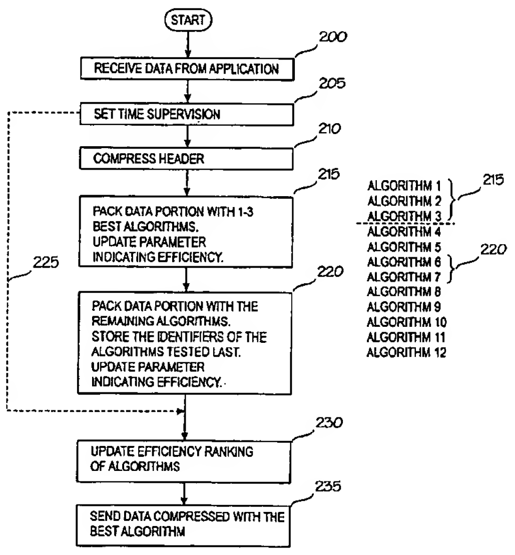
* cited by examiner

Primary Examiner—Wellington Chin
Assistant Examiner—Brenda Pham
(74) *Attorney, Agent, or Firm*—Pillsbury Winthrop LLP

(57) **ABSTRACT**

The invention relates to compressing and transmitting data on a connection between two parties in a telecommunication system comprising at least one slow transmission channel, such as the air interface Um of the radio network. The data to be transmitted are assembled into frames (F) comprising a header section (1) and a data section (2). Prior to transmission, at least the header (1) or the data section (2) of at least some of the frames (F) are compressed. The transmitting party has available at least two different compression algorithms and the receiving party has available at least two different decompression algorithms. The transmitting party compresses at least one section (1, 2) of at least some of the frames (F) with at least two different algorithms, and transmits the frame (F) compressed with the algorithm that produced the best compression ratio.

17 Claims, 2 Drawing Sheets



START
200
RECEIVE DATA FROM APPLICATION
205
SET TIME SUPERVISION
210
COMPRESS HEADER
215
PACK DATA PORTION WITH 1-3 BEST ALGORITHMS. UPDATE PARAMETER INDICATING EFFICIENCY.
225
220
PACK DATA PORTION WITH THE REMAINING ALGORITHMS. STORE THE IDENTIFIERS OF THE ALGORITHMS TESTED LAST. UPDATE PARAMETER INDICATING EFFICIENCY.
230
UPDATE EFFICIENCY RANKING OF ALGORITHMS
235
SEND DATA COMPRESSED WITH THE BEST ALGORITHM
ALGORITHM 1
ALGORITHM 2
ALGORITHM 3
ALGORITHM 4
ALGORITHM 5
ALGORITHM 6
ALGORITHM 7
ALGORITHM 8
ALGORITHM 9
ALGORITHM 10
ALGORITHM 11
ALGORITHM 12
215
220

Fig. 1

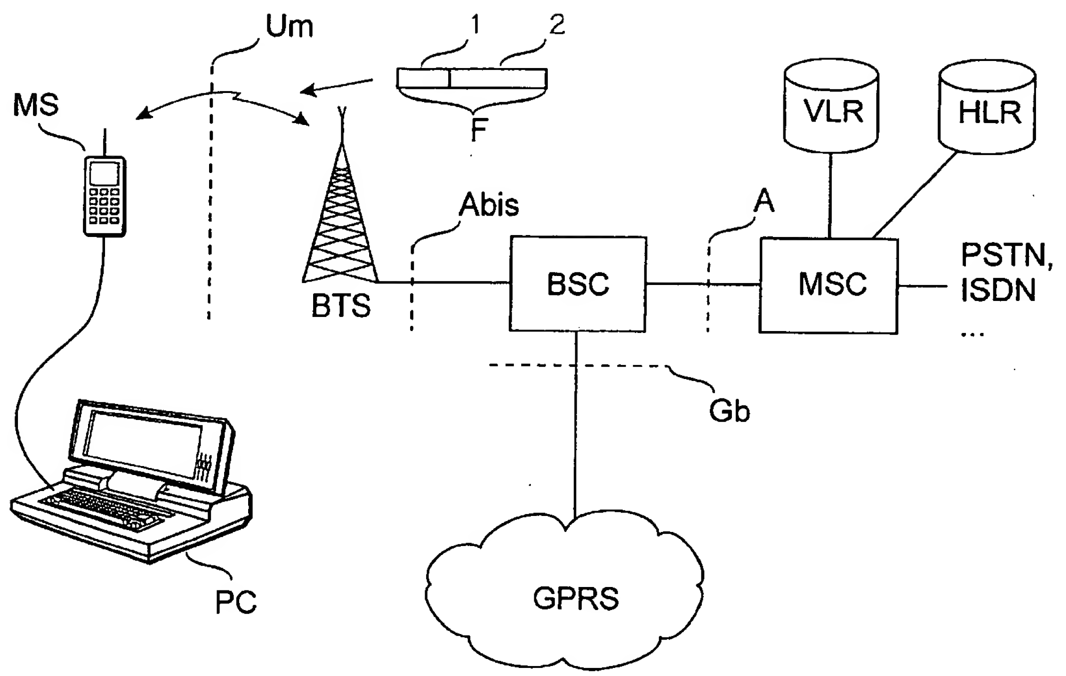


Fig. 2A

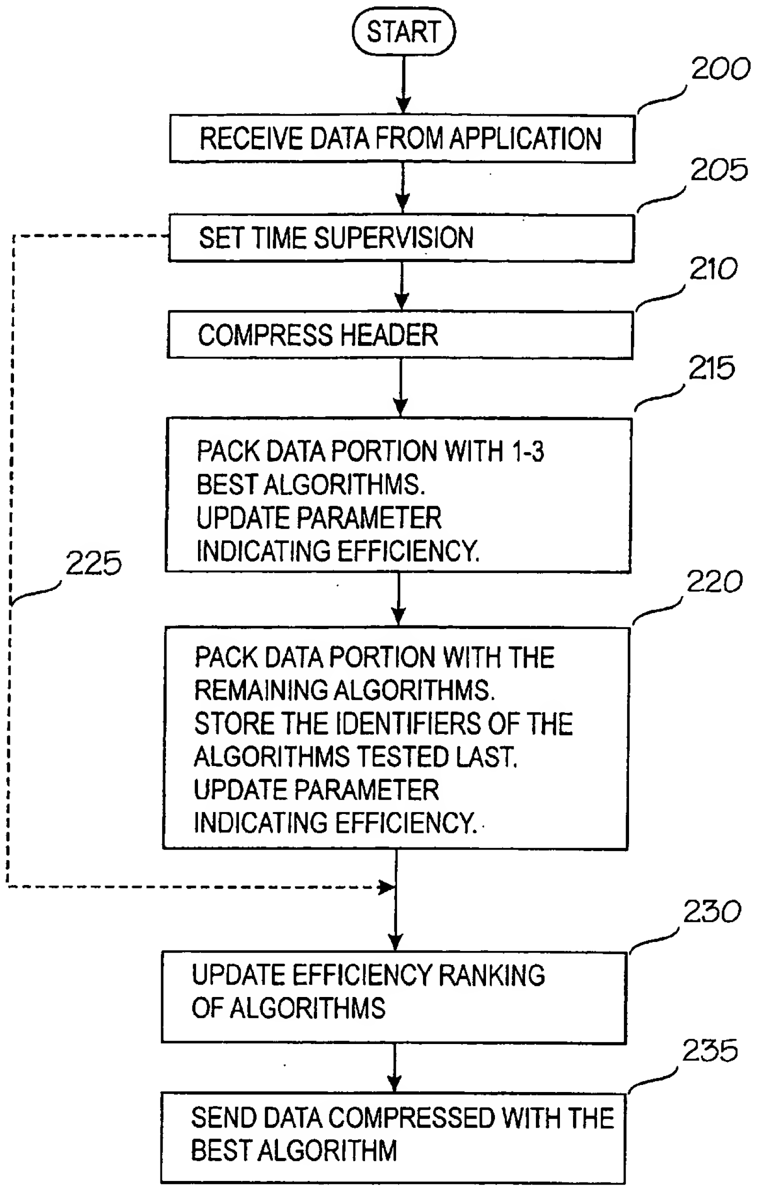
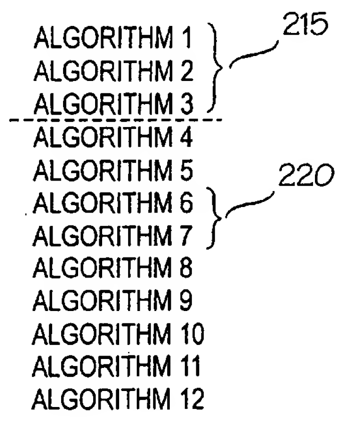

START
200
RECEIVE DATA FROM APPLICATION
205
SET TIME SUPERVISION
210
COMPRESS HEADER
215
PACK DATA PORTION WITH 1-3 BEST ALGORITHMS. UPDATE PARAMETER INDICATING EFFICIENCY.
225
220
PACK DATA PORTION WITH THE REMAINING ALGORITHMS. STORE THE IDENTIFIERS OF THE ALGORITHMS TESTED LAST. UPDATE PARAMETER INDICATING EFFICIENCY.
230
UPDATE EFFICIENCY RANKING OF ALGORITHMS
235
SEND DATA COMPRESSED WITH THE BEST ALGORITHM

Fig. 2B


ALGORITHM 1
ALGORITHM 2
ALGORITHM 3
215
ALGORITHM 4
ALGORITHM 5
ALGORITHM 6
ALGORITHM 7
220
ALGORITHM 8
ALGORITHM 9
ALGORITHM 10
ALGORITHM 11
ALGORITHM 12

DATA COMPRESSION ON A DATA CONNECTION

This application is the national phase of international application PCT/F197/00345 filed Jun. 3, 1997 which designated the U.S.

BACKGROUND OF THE INVENTION

The invention relates to improving the capacity of data transfer in a communication system, or more specifically a mobile communication system.

FIG. 1 shows, from the point of view of the invention, the essential parts of a cellular mobile communication system. Mobile stations (MS) communicate with base transceiver stations (BTS) over an air interface Um. The base stations are controlled by base station controllers (BSC) which are connected to mobile switching centers (MSC). A subsystem under control of a base station controller BSC, including the base stations BTSn it controls, is commonly referred to as a base station subsystem (BSS). The interface between the mobile switching center MSC and the base station subsystem BSS is referred to as an A-interface. The part of the mobile communication system at the MSC side of the A-interface is referred to as a network subsystem (NSS). Correspondingly, the interface between the BSC and the BTS is referred to as an Abis interface. The mobile switching center MSC handles the connecting of incoming and outgoing calls. It performs functions similar to those of an exchange of a public switched telephone network (PSTN). In addition to these, it also performs functions characteristic of mobile communications only, such as subscriber location management, jointly with the subscriber registers VLR and HLR of the network. As an alternative to the circuit switched connection described above, the connection to the mobile station MS may also take place via a packet network GPRS (General Packet Radio Service).

One of the growing fields of application for mobile stations is to establish data links in connection with portable computers. Such a computer is represented by the computer PC in FIG. 1. The computer PC and the mobile station MS may be separate units or they may form an integrated whole. The data to be transmitted on data links is assembled into frames (F) that typically contain a header section 1 and a data section 2. If an increase occurs in the number and/or use of mobile stations, a bottleneck will be met in the form of the transfer capacity of the air interface Um. The transfer capacity of the air interface may be increased by compressing the data to be transmitted over the air interface. The compression is based on some bit patterns in the data stream being frequent and some occurring only once. The bit patterns which occur frequently may be sent only once as a whole, and later it is possible to send only a reference to the bit pattern sent earlier. A number of such compression algorithms has been developed,-including RLE (Run Length Encoding) and LZ (Lempel-Ziv) with its different variants, JPEG, MPEG etc. Within the scope of this application, compression ratio of an algorithm refers to the ratio between the length of an uncompressed bit string (e.g. a frame) and the length of the bit string compressed with the algorithm in question.

The conventional packet communication described above encounters the problem that data of a specific type may be compressed more efficiently with a particular compression algorithm whereas data of some other type may be compressed more efficiently with another algorithm. Also such bit patterns exist that cannot be compressed with any

algorithm, whereby the identifier indicating the algorithm used just adds to the length of the bit string. A further problem with the prior art compression carried out with a fixed algorithm is that is does not offer optimal protection against eavesdropping because the algorithm does not change.

For example, PCT application WO 94/14273 discloses a system for compressing data to be transmitted with a number of different compression means. However, the WO 94/14273 application mainly relates to transmission of video information to several receivers simultaneously. Because the same information is transmitted to several receivers, such a system does not comprise a bottleneck comparable to the air interface of a cellular packet radio network wherein unique information is transmitted between each pair of transmitter and receiver. Also the WO 94/14273 application assumes that the best compression method can be determined by the contents of the transmitted information. This assumption may be correct if it is known before transmission that the information will be video information.

BRIEF SUMMARY OF THE INVENTION

It is consequently the object of the invention to develop a method by means of which the limited capacity of the air interface or another low-speed telecommunication resource may be utilized as efficiently as possible, thereby simultaneously enhancing traffic encryption against unauthorized listening. The objects of the invention are achieved by a method which is characterized by that which is set forth in the independent claims. The preferred embodiments of the invention are set forth in the dependent claims.

The invention is based on the notion that in a general case, when the contents of the data to be transmitted may be arbitrary, the most efficient compression algorithm can be established only experimentally. Because of this, the data to be transmitted are, according to the invention, compressed with a number of different algorithms, and the best of the compression results is transmitted to the receiving party. The invention is further based on the view that it is worth while in a telecommunication system which contains a slow telecommunication resource, such as an air interface limiting the capacity, to carry out a lot of extra computation in a location where that is cheap and where capacity enlargements may result from such an action.

The method according to the invention utilises the bandwidth of the bottleneck (such as the air interface in a telecommunication system) in the most effective manner possible because the compression algorithm is selected on the basis of actual testing between different algorithms. The method is applicable to several types of telecommunication systems. The method according to the invention may be used adaptively so that the transmitting party learns to apply a specific algorithm on a specific connection. A further advantage of the inventive method is provided by improvements regarding the protection of telecommunication as an eavesdropper will have difficulties in interpreting a message whose compression algorithm may change in the middle of the connection, even between successive packets.

BRIEF SUMMARY OF THE FIGURES

In the following, the invention will be described in closer detail in connection with its preferred embodiments and with reference to the accompanying drawings in which:

FIG. 1 shows the parts of the mobile communication network that are significant to the invention,

FIGS. 2A and 2B illustrate the steps of the inventive compression.

DETAILED DESCRIPTION OF THE INVENTION

With reference to FIG. 1, the invention will be described in an environment in which there is a mobile station MS and a computer PC connected thereto at the first end of the transmission channel, and at the other end a network subsystem NSS of which is shown a base station BTS, a base station controller BSC and a mobile switching center MSC. To keep the description simple, it is assumed that the compression and decompression algorithm according to the invention is implemented in the MS, but it may equally well be implemented in the PC. The preferred embodiment of the invention relates to enhancing the traffic that takes place over the air interface Um, but in place of the air interface there may also be some other transmission channel having a limited capacity.

In its simplest form, the invention may be applied so that a fixed set of compression and decompression algorithms are implemented in the mobile station MS and the network subsystem NSS. This set of algorithms is the same in the MS and the NSS. The transmitting party compresses each frame F with each algorithm it knows, and selects the algorithm that yields the best compression ratio. In addition, the transmitting party inserts an information field in the frame to be transmitted, indicating which algorithm the data have been compressed with. On the basis of this information field, the receiving party will know which algorithm to use in order to decompress the data.

This simple implementation has at least two problems. Firstly, when using different equipments, the transmitting and receiving parties do not know in advance which algorithms have been installed at the counterpart. In addition, it is not possible to add new algorithms in the system without simultaneous updating of all the equipments in the system. Secondly, compressing each frame with all possible algorithms considerably increases requirements for computation capacity. The first of the problems may be solved by a negotiation protocol between the transmitting and receiving parties. The latter problem may be solved by optimizing the compression procedure, e.g. by making it adaptive.

In a mobile communication system, it could in principle be speculated that the negotiation for finding out the capabilities of the parties is unnecessary and that the features of each mobile station could be stored in a home location register or a similar equipment register. This arrangement would involve a lot of problems, the biggest of which probably having to do with the GSM type of systems maintaining the subscriber location by means of SIMs (Subscriber Identity Modules). If a SIM is shifted to another, for example a rented, terminal equipment, the system will assume that the second terminal equipment has the same capabilities as the original one. This assumption is not necessarily correct.

For this reason, it is better to identify the capabilities of the transmitting and receiving parties at the beginning of the communication. In mobile communication systems, the negotiation must be carried out anew when the mobile station roams to a new area. A suitable and simple negotiation protocol is, for example, such that when taking new algorithms in use they are issued a running number, and at the beginning of the connection or in association with a location area change the calling equipment transmits an inquiry to the called equipment, concerning the available algorithms, to which the called equipment responds by sending a bitmap in which a bit set at an algorithm means that the algorithm is available. According to an alternative

negotiation protocol, the calling equipment transmits a short test message compressed with all the algorithms being tested, and the called equipment responds with an affirmative acknowledgment if it is capable of decompressing the message, and if it is not, with a negative acknowledgment. If there is installed, for each compression algorithm, both a compressing and a decompressing algorithm in the transmitting party as well as the receiving party, the common capabilities of the transmitting party and the receiving party may be identified so that e.g. the calling equipment identifies the capabilities of the called equipment. It is unnecessary for the called equipment to identify the capabilities of the calling equipment as the latter cannot have capabilities other than those it has already inquired of the called equipment.

On the other hand, it is mathematically considerably simpler to decompress a packet than to compress it. This holds particularly well true regarding compression of live video images and fractal algorithms, especially. Consequently, it could also be conceivable to implement e.g. in mobile stations a larger number of algorithms for decompressing a packet than for compressing it. In such a case, the negotiation protocol may be supplemented so that also the called equipment identifies the capabilities of the calling equipment.

For an efficient compression of data, the compression must take place prior to encryption and splitting the data into frames. According to an embodiment of the invention, the header sections 1 of the frames F are compressed with a specific type of algorithm and the data sections 2 of the frames F are compressed with at least two different algorithms of which the one is selected that provides the best compression ratio. The header section is compressed with an algorithm optimized for this purpose, said algorithm being substantially the same from one frame to another.

For example, when transmitting user-entered characters in TCP/IP frames in a Telnet connection, each character entered is normally sent in its own frame. This is because the user awaits a relatively quick response, after no more than approximately 0.2 seconds from entering the character. On the other hand, the transmitting program does not know at which moment the next entering is to take place, which means that each character has to be sent as a separate frame consisting of 40 bytes of header and 1 byte of data. The reference [1] discloses a method for compressing the header sections of TCP/IP frames into 3–5 bytes. The method according to the reference [1] utilizes the fact that in the header sections of TCP/IP frames a lot of information is sent that the receiving party is able to form by itself on the basis of the header section in the previous frame. Similar redundancy is present e.g. in header sections of ATM cells.

The compression algorithm employed may be indicated to the receiving party for example by transmitting, in each compressed frame F, an identifier indicative of the compression algorithm used. Alternatively, the receiving party may be sent information on the change in the compression algorithm. This information may be a separate frame, or a field or a peculiar bit pattern inserted to a frame normally transmitted.

To utilize the air interface as efficiently as possible, it is advantageous to carry out the compression procedure during those moments when the transmitting party for a reason or another may not transmit. In the GSM system, for example, the data are transmitted over the air interface as bursts in TDMA timeslots. Between the bursts, the transmitting party can perform the necessary computations and thus utilize its computation capacity that it would otherwise waste just waiting for the next timeslot. At the allocated timeslot, it is possible to interrupt the compression and to transmit the frame compressed with the algorithm that up to that point has produced the best result.

The following discusses various options for reducing the computation requirements. The compression procedure can be optimized by for example compressing a frame F with all the algorithms used, i.e. those that were in the negotiations found to be supported by both parties.

The frame F that is compressed with all the algorithms employed is preferably the first one representing a specific type of data, i.e. when the data type changes, the testing with the algorithms may be started from the beginning. The subsequent frames are each first compressed with the algorithm that at the previous frame yielded the best result. The other compression algorithms are not necessarily completed. Their processing may be interrupted immediately as the length of the compressed frame obtains the length of the algorithm compressed with the optimal algorithm.

According to another embodiment, a sliding average of the compression ratio is maintained-for each algorithm with which a frame has been compressed. The frames are compressed with different algorithms in an order determined by the compressing ratio obtained with the algorithms. If the compression ratio of a frame with some algorithm exceeds a predetermined threshold value, the frame in question will not be compressed with the other algorithms. The predetermined threshold value may be a fixed multiplier, for example a number between 3–6. Alternatively, the threshold value may be a specific percentage, for example 80–90% of the compression ratio achieved with the best algorithm up to that point.

If the time available is not sufficient for compressing each frame with each algorithm, the individual frames may be compressed for example with those 1–3 algorithms that up to that point show the best sliding average for the compression ratio. The remaining time—e.g. when waiting for the transmission turn—is spent on testing the remaining algorithms one at a time. This is illustrated by the following example. It is assumed that there are 7 different algorithms available, and when waiting for each transmission timeslot there is time to test four of them. In such a case, at each timeslot, compression may be carried out with those two algorithms that up to that point have yielded the best result. Of the other five algorithms, two are tested at each timeslot.

Encryption may further be improved if the algorithms are during the negotiation numbered in an order determined on the basis of a pseudorandom number known only to the parties. The same outcome may be reached by changing the identifiers of the algorithms, at a specific algorithm during the connection. It is advantageous to carry out the negotiation in an encrypted form. With these methods the advantage is obtained that the compression algorithm identifiers transmitted along with the frames are not unequivocal to eavesdroppers.

FIGS. 2A and 2B illustrate some of the steps in the compression method of the invention. At step **200**, the compression routine receives the data to be transmitted and compressed from the application producing them. Step **205** illustrates the operations in which time supervision is set to ensure that the data are transmitted on time, for example in the next transmission timeslot. At step **210**, the header **1** of the frame F is compressed with an algorithm optimized for this purpose. At step **215**, the data section **2** is compressed by a few, in this case three, of the best algorithms up to that point. In association with the compression, a parameter is

maintained for each algorithm, illustrating its efficiency, such a parameter being for example the sliding average of the compression ratio. At step **220**, the data section **2** is compressed with as many of the remaining algorithms as possible within the time supervision, in this example with two algorithms. The identifiers of the algorithms tested at this step are stored in memory, and next time at step **220**, other algorithms are tested. The testing may be begun with the algorithm that follows the one tested last. When the time supervision **225** expires, for example when a transmission turn approaches, the order from best to worst of the algorithms is updated at step **230**, and the data are transmitted at step **235** compressed with the algorithm that produced the best result within the time available.

The preferred embodiment of the invention relates to compressing data on a communication link and particularly in a packet radio network. The invention is applicable for use in other types of telecommunication systems and in data processing systems which have at least one distinct bottleneck restricting the capacity of the entire system. Therefore, it is obvious for a person skilled in the art that upon advancements in technology the basic idea of the invention may be implemented in many ways. The invention and its embodiments are consequently not restricted to the examples described above but they may vary within the scope of the claims.

References

[1] V. Jacobson: Compressing TCP/IP headers for low-speed serial links (Request For Comments 1144).

What is claimed is:

1. A method for compressing and transmitting data on a connection between two parties in a telecommunication system comprising at least one slow transmission channel (Um), the method comprising:
   - assembling the data to be transmitted into frames (F) which contain at least a header section (1) and a data section (2), and
   - compressing at least one section (1, 2) of at least some of the frames (F) prior to transmission,
   - making at least two different compression algorithms available to the transmitting party,
   - making at least two different decompression algorithms available to the receiving party,
   - characterized in that the transmitting party:
   - compresses at least one section (1, 2) of at least some of the frames (F) with at least two different compression algorithms;
   - selects the compression algorithm that yielded the best compression ratio; and
   - transmits the frame (F) over the slow transmission channel (Um) to the receiving party, compressed with said selected compression algorithm.

2. A method as claimed in claim **1**, characterized in that the parties negotiate the compression algorithms to be used on the connection at least at the beginning of the connection.

3. A method as claimed in claim **2**, characterized by the first party sending to the second party an inquiry concerning the available algorithms, to which the second party responds by sending a list containing information on the algorithms available to the second party.

4. A method as claimed in claim **2**, characterized by the second party selecting the algorithms that both the parties support from the list sent by the first party.

5. A method as claimed in claim **2**, characterized by the second party transmitting a list of the algorithms available to it regardless of which algorithms the first party has available.

6. A method as claimed in claim **2**, characterized by the first party transmitting a brief test message compressed separately with each algorithm being tested, and the second party responding with an affirmative acknowledgment if it is capable of decompressing the message, and if not, with a negative acknowledgment.

7. A method as claimed in claim **2**, characterized in that both parties have a decompression algorithm available for substantially each compression algorithm.

8. A method as claimed in claim **2**, characterized in that at least one of the parties may have available a decompression algorithm without a corresponding compression algorithm or vice versa, and that the parties negotiate separately the compression and decompression algorithms available to them.

9. A method as claimed in claim **2**, characterized in that the negotiation between the parties is transmitted on an encrypted channel.

10. A method as claimed in claim **2**, characterized in that the identifiers of the compression algorithms are changed between successive connections and/or during one connection.

11. A method as claimed in claim **1**, characterized by

compressing the header sections (1) of the frames (F) with an algorithm which is substantially the same between two successive frames (F), and

compressing the data sections (2) of the frames (F) with at least two different algorithms of which the one is selected that yields the best compression result.

12. A method as claimed in claim **1**, characterized by transmitting, with each compressed frame (F), an identifier indicative of the compression algorithm used.

13. A method as claimed in claim **1**, characterized in that when the compression algorithm changes the transmitting party separately transmits information on the change to the receiving party.

14. A method as claimed in claim **1**, characterized by

compressing a frame (F), which is advantageously the first one representing a specific type of data, with substantially all the algorithms known to both parties,

compressing the subsequent frames (F) first with the algorithm that at the previous frame (F) produced the best result,

discontinuing to carry out the other algorithms if the length of the frame (F) compressed with the algorithm in question exceeds the length of the frame (F) compressed with up to that point the best algorithm.

15. A method as claimed in claim **1**, characterized by the transmitting party being assigned a restricted time for the compression, such as the time between two successive transmit turns in time division multiple access systems.

16. A method as claimed in claim **15**, characterized in that at the end of the restricted time the transmitting party discontinues testing the compression algorithms and transmits the frame (F) compressed with the algorithm that up to that point has yielded the best compression result.

17. A method as claimed in claim **16**, characterized by

maintaining, for the different compression algorithms, a parameter indicating efficiency, advantageously a sliding average of the compression ratio,

compressing each frame (F) or a section (**1, 2**) thereof with a few, advantageously 1–3 of the best compression algorithms up to that point, in an order determined by the parameter indicating efficiency, and

testing, in the remaining part of the restricted time, the remaining algorithms alternately so that at successive frames (F) different algorithms are tested.

* * * * *

US006542931B1

(12) **United States Patent**
Le et al.

(10) **Patent No.: US 6,542,931 B1**
(45) **Date of Patent: Apr. 1, 2003**

(54) USING SPARSE FEEDBACK TO INCREASE BANDWIDTH EFFICIENCY IN HIGH DELAY, LOW BANDWIDTH ENVIRONMENT

(75) Inventors: **Khiem Le**, Coppell, TX (US); **Haihong Zheng**, Coppell, TX (US); **Zhigang Liu**, Irving, TX (US); **Christoph Clanton**, Richardson, TX (US)

(73) Assignee: **Nokia Corporation**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/434,384**

(22) Filed: **Nov. 5, 1999**

(51) **Int. Cl.**7 .... **G06F 15/16**; H04L 9/00

(52) **U.S. Cl.** **709/228**; 709/217; 709/232; 709/247; 713/160; 714/750; 341/60

(58) **Field of Search** 714/749, 750; 709/227, 247, 217, 228, 232; 341/60; 713/160

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,797,881 A	*	1/1989	Ben-Artzi	370/402
4,896,151 A	*	1/1990	Kuranami et al.	714/748
5,528,605 A	*	6/1996	Ywoskus et al.	370/507
5,684,791 A		11/1997	Raychaudhuri et al.	
5,872,777 A		2/1999	Brailean et al.	
5,970,063 A	*	10/1999	Chapman et al.	370/346
6,032,197 A	*	2/2000	Birdwell et al.	709/247

OTHER PUBLICATIONS

Wei, et al. "Improvement of TCP Performance on Asymmetric Channels Based on Enhancements of Compressing TCP/IP Header Algorithm", International Conference on Communication Technology (ICCT '98), Oct. 22–24, 1998, Beijing, China.*

W. Zwaenepoel, "Protocol for Large Data Transfers over Local Networks" Proceedings of the Data Communications Symposium, U.S. Washington, IEEE Computer Society Press, vol. SYM. 9, Sep. 1, 1985, pp. 22–32.

S. Bakhtiyari et al, "A Robust Type II Hybrid ARQ Scheme with Code Combining for Mobile Communications" Proceedings of the Pacific RIM Conference on Communications, Computers and Signal Processing, May 19, 1993, pp. 214–217.

* cited by examiner

Primary Examiner—David Wiley
Assistant Examiner—George C Neurauter
(74) *Attorney, Agent, or Firm*—Antonelli, Terry, Stout & Kraus, LLP

(57) **ABSTRACT**

A method and apparatus for eliminating the inefficient use of network bandwidth cause by numerous acknowledgment transmitted by the receiver to the transmitter by providing sparse feedback from the receiver to the transmitter indicating receipt of packets having headers to be used as reference headers. In the invention, upon receipt in the receiver of a packet having a reference header, a feedback is provided to the transmitter indicating receipt of the packet having the reference header. Thereafter, the receiver waits a predetermined period of time before providing another feedback in response to another packet having a reference header. The predetermined period of time allows time for the feedback to be received by the transmitter and for information from the transmitter indicating receipt of the feedback to be received by the receiver.

10 Claims, 4 Drawing Sheets

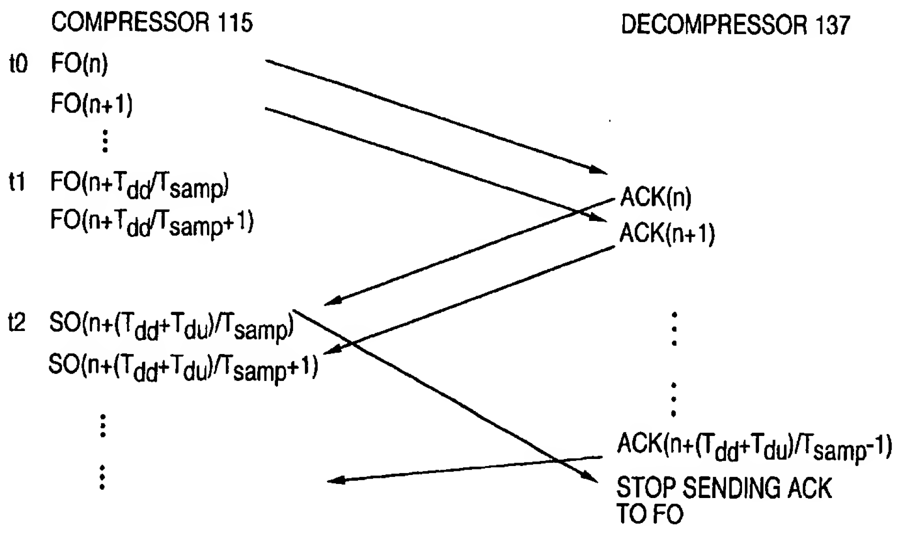


FIG. 1

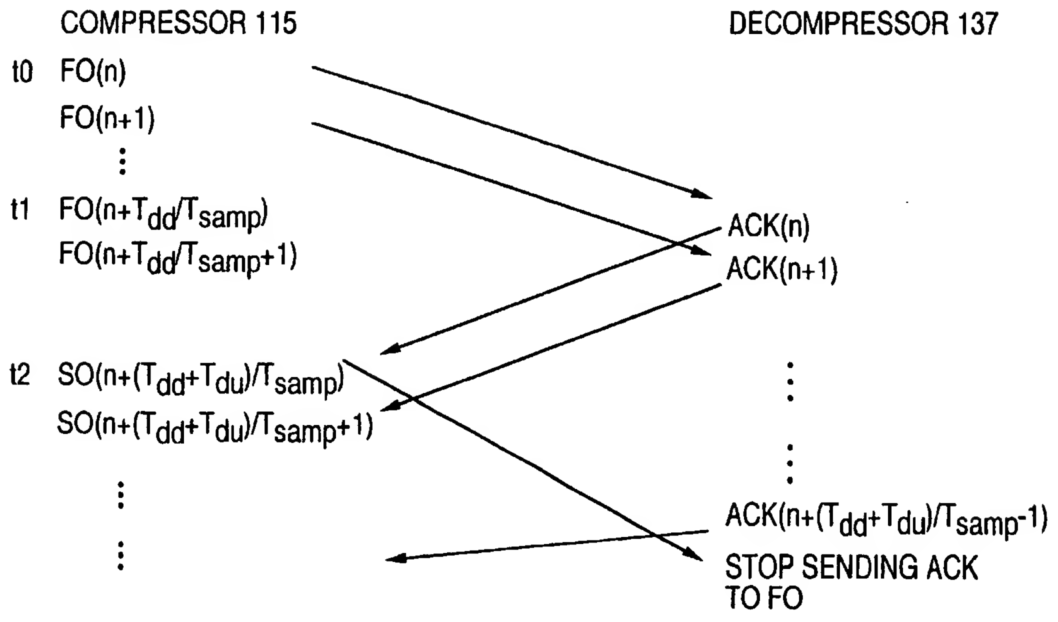
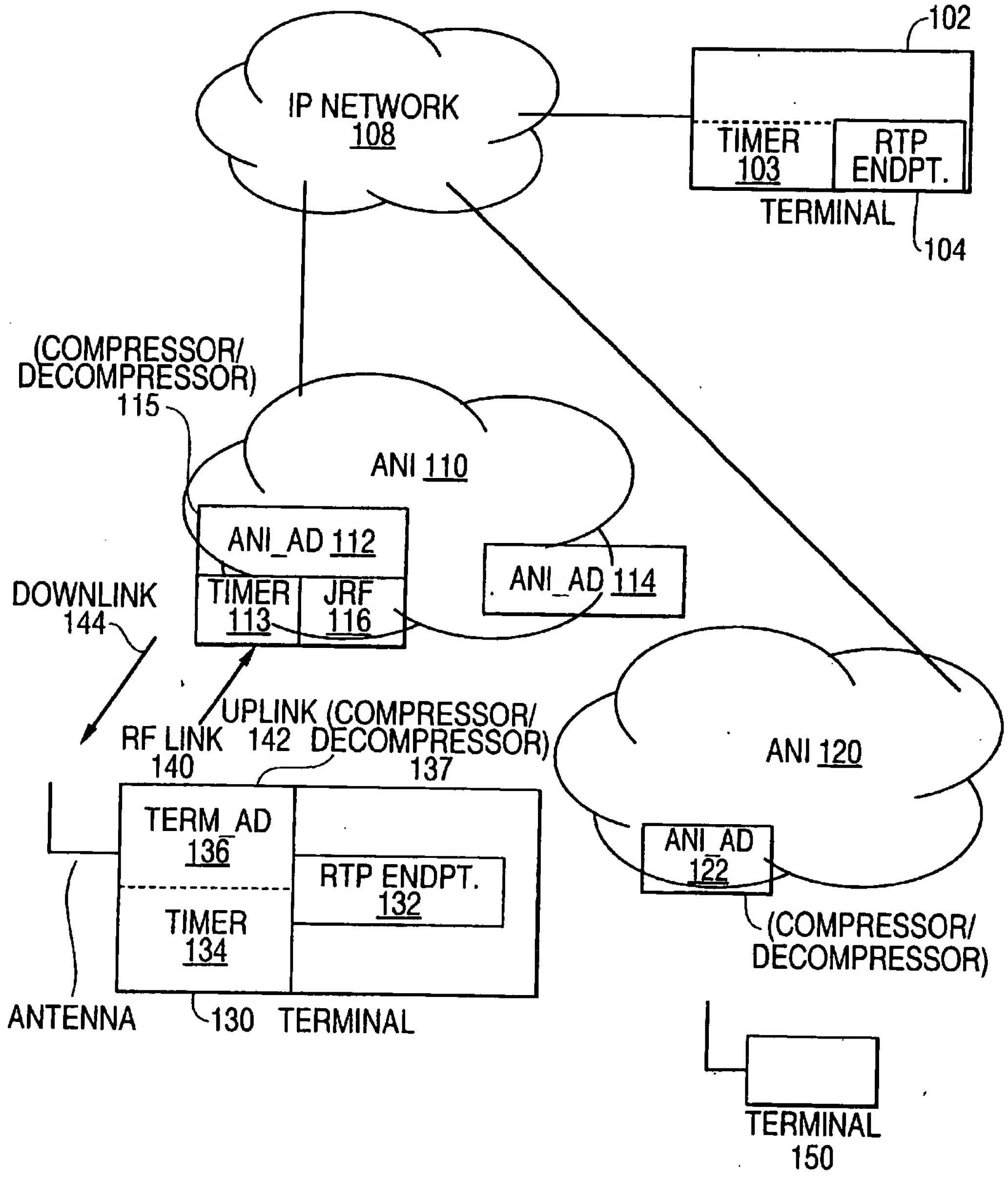

COMPRESSOR 115
DECOMPRESSOR 137
t0 FO(n)
FO(n+1)
t1 $FO(n+T_{dd}/T_{samp})$
$FO(n+T_{dd}/T_{samp}+1)$
t2 $SO(n+(T_{dd}+T_{du})/T_{samp})$
$SO(n+(T_{dd}+T_{du})/T_{samp}+1)$
ACK(n)
ACK(n+1)
$ACK(n+(T_{dd}+T_{du})/T_{samp}-1)$
STOP SENDING ACK TO FO

FIG. 2



102
IP NETWORK 108
TIMER 103
RTP ENDPT.
TERMINAL
104
(COMPRESSOR/ DECOMPRESSOR) 115
ANI 110
ANI_AD 112
TIMER 113
JRF 116
ANI_AD 114
DOWNLINK 144
UPLINK 142
RF LINK 140
(COMPRESSOR/ DECOMPRESSOR) 137
ANI 120
ANI_AD 122
(COMPRESSOR/ DECOMPRESSOR)
TERM_AD 136
TIMER 134
RTP ENDPT. 132
ANTENNA
130 TERMINAL
TERMINAL 150

FIG. 3

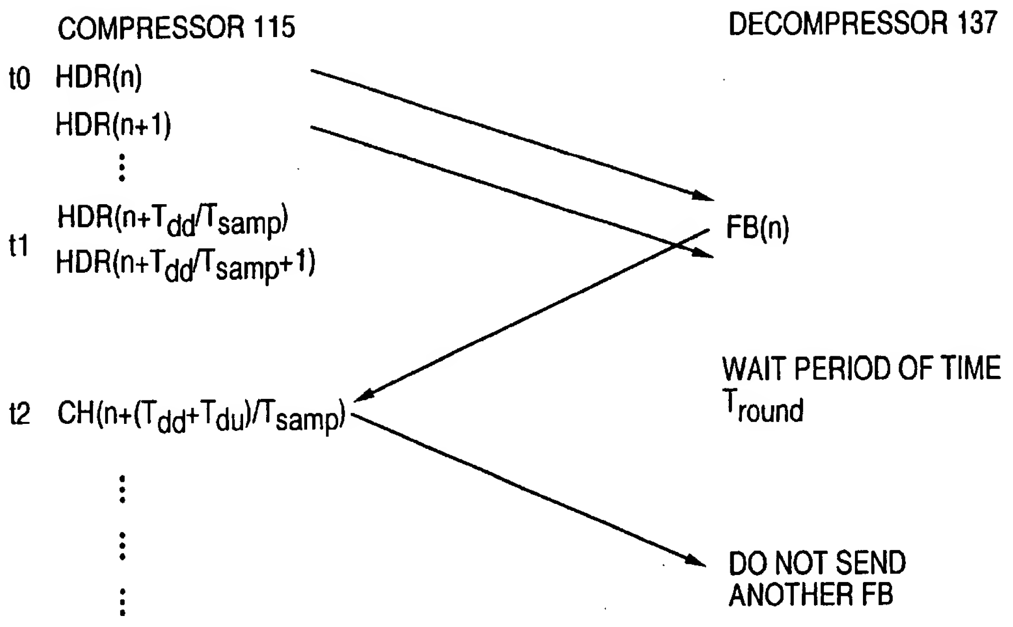
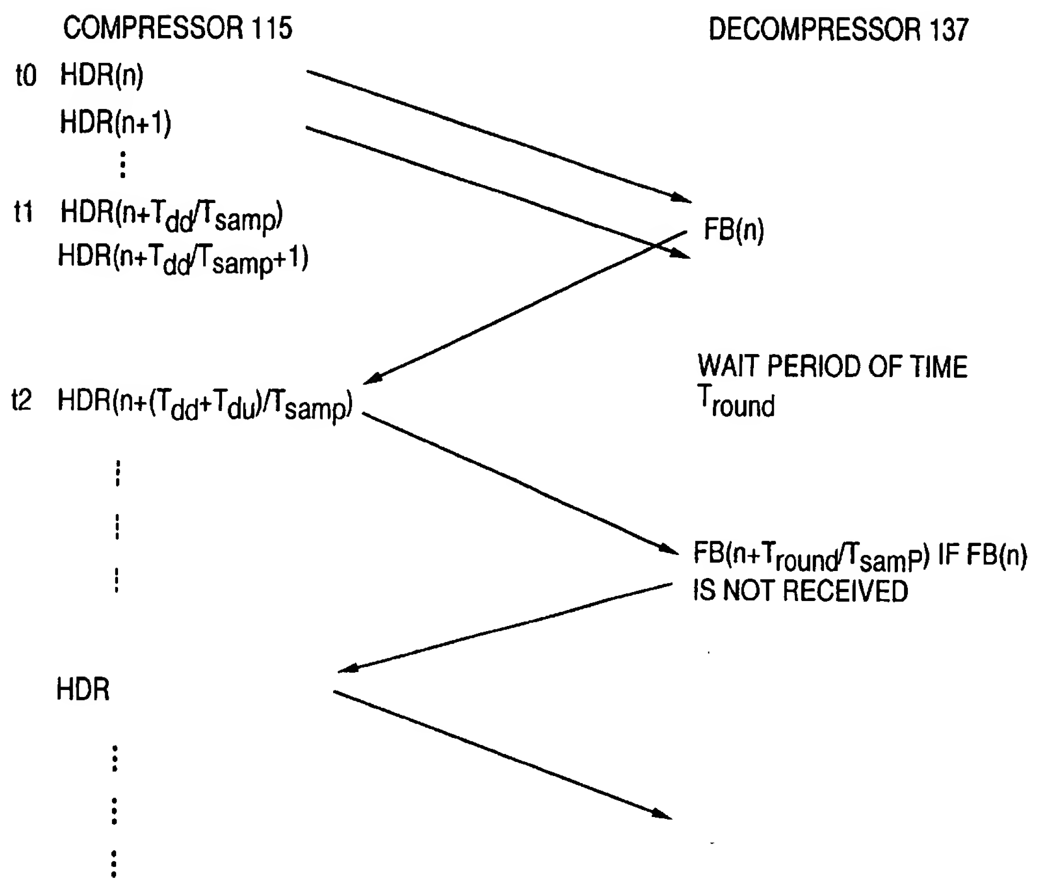

COMPRESSOR 115
DECOMPRESSOR 137
t0 HDR(n)
HDR(n+1)
HDR($n+T_{dd}/T_{samp}$)
t1
HDR($n+T_{dd}/T_{samp}+1$)
FB(n)
WAIT PERIOD OF TIME T_{round}
t2 CH($n+(T_{dd}+T_{du})/T_{samp}$)
DO NOT SEND ANOTHER FB

FIG. 4


COMPRESSOR 115
DECOMPRESSOR 137
t0 HDR(n)
HDR(n+1)
t1 HDR(n+T_{dd}/T_{samp})
HDR(n+T_{dd}/T_{samp}+1)
FB(n)
WAIT PERIOD OF TIME T_{round}
t2 HDR(n+(T_{dd}+T_{du})/T_{samp})
FB(n+T_{round}/T_{samP}) IF FB(n) IS NOT RECEIVED
HDR

USING SPARSE FEEDBACK TO INCREASE BANDWIDTH EFFICIENCY IN HIGH DELAY, LOW BANDWIDTH ENVIRONMENT

BACKGROUND OF THE INVENTION

The present invention relates to a method and apparatus for eliminating the inefficient use of network bandwidth caused by numerous acknowledgements transmitted by a receiver to a transmitter by providing sparse feedback from the receiver to the transmitter to indicate receipt of packets having headers to be used as reference headers.

For Internet Protocol (IP) based real-time multimedia applications packets are used to carry the real-time data. Each packet includes a header and a payload. The header carries information such as source and destination addresses of the packet and the payload carries the data to be transmitted. Each packet is formatted according to the IP and Real-time Transfer Protocol (RTP) which is predominately used on top of User Datagram Protocol (UDP). RTP is described in detail in "RTP: A Transport Protocol for Real-Time Applications" by H. Schulzrinne, et al, Internet Engineering Task Force (IETF) Request for Comments (RFC) 1889, January 1996. The size of a combined IP/UDP/RTP header for a packet is at least 40 bytes for IPv4 and at least 60 bytes for IPv6. A total of 40–60 bytes of overhead per packet may be considered heavy in systems (e.g., such as cellular networks) where spectral efficiency is a common concern. Consequently, a need arises for suitable IP/UDP/RTP header compression mechanisms.

A current header compression scheme is described in "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links" by S. Casner et al, IETF, RFC 2508, February 1999 and "IP Header Compression" by M. Degermark, et al, IETF, RFC 2507, February 1999. The header compression scheme described in RFC 2508 is able to compress the 40-60 bytes IP/UDP/RTP header down to 2 or 4 bytes over point-to-point links. This header compression scheme is based on the observation that most fields of the headers of the packets remain constant in a packet stream during the length of a session. Thus, it is possible to compress the header information by establishing a compression state (context) at a compressor (transmitter) and a decompressor (receiver). Packets having compressed headers are then transmitted from the compressor to the decompressor, wherein the compressed headers correspond to a reference header stored as part of the compression state. Each compressed header contains a minimum amount of information. The information carried in the compressed header is decompressed at the decompressor based on the established compression state.

In RFC 2508 the changes occurring in the RTP header fields from one packet to the next, such as the RTP time stamp, can be predicted by linear extrapolation from the preceding header which was received without error. Thus, in an RTP header the primary information that is sent is a sequence number which is used for packet loss detection. To initiate a session or to re-synchronize a compression state between a compressor and decompressor, a packet containing a Full Header (FH) is transmitted from the compressor to the decompressor. The FH contains all of the information of the header of the packet and is used (stored) as a reference header. After a session has been initialized or re-synchronization has been performed, all subsequent packets are transmitted with compressed headers. The compressed headers are, for example, of two types.

The first type of compressed header is used when the subsequent headers of the subsequently transmitted packets can be extrapolated in a linear fashion from the previous header. In this setting, the compressor transmits sequence numbers as the compressed headers. This type of compressed header is referred to as Second Order (SO) header. The second type of compressed header is used when the subsequent headers of the subsequently transmitted packets cannot be extrapolated in a linear fashion. In this setting, the compressor transmits additional information including the sequence number as the compressed headers. This type of compressed header is referred to as a First Order (FO) header. The FO header contains additional information which is required to accurately decompress the compressed headers of the subsequently transmitted packets. In RFC 2508, all headers that are decompressed are stored as reference headers. In order to decompress the current header, the decompressor must have correctly decompressed the previous header. In practice, it is not uncommon for packets having compressed header to be lost or corrupted during transmission. This results in the compressor and decompressor staying in a less then optimal state for some extended time. The same can occur due to Round Trip delays in receiving the packets having compressed headers. Consequently, the data streams being processed by the compressor and decompressor may require additional bandwidth.

To improve the header compression scheme described in RFC 2508, the decompressor transmits acknowledgements to the compressor indicating receipt of a FH or FO header packet. The compressor in response to an acknowledgement indicating receipt of a FH packet switches to an FO state and begins transmitting FO header packets where linear extrapolation cannot be conducted or switches to the SO state and begins transmitting SO header packets where linear extrapolation can be conducted. Similar to an FH packet the compressor, in response to an acknowledgment indicating receipt of an FO header packet, switches to the SO state and begins transmitting SO header packets.

In error/loss prone communication environments, such as cellular, the decompressor cannot be certain that the acknowledgment has been properly received by the compressor until it sees an altered behavior on the part of the compressor. Namely, the decompressor in conventional apparatus is not aware that the acknowledgment has been properly received, until it sees that the compressor has altered its behavior. That is, the decompressor sees FO header packets instead of FH packets or SO header packets instead of FO header packets. In the mean time, the decompressor keeps sending acknowledgments to headers of each of the packets received. Further, the decompressor in conventional apparatus remains unaware that the acknowledgment has been properly received due to round trip delays in receiving packets resulting rom the altered behavior of the compressor. Thus, the conventional technique suffers from the disadvantage of inefficient use of bandwidth of the network.

FIG. 1 graphically illustrates the inefficient use of the bandwidth of a network resulting from sending acknowledgments in response to each and every FH packet or FO header packet even after a first acknowledgment has been sent. According to the above, as each FH packet or FO header packet is received an acknowledgment is transmitted from the decompressor to the compressor acknowledging receipt of the FH packet or FO header packet. In FIG. 1, it is assumed that at time t_0 an FO(n) header packet is transmitted from the compressor and detected by the decompressor at time t_1. Further, at time t_1 the decompressor, in response to the FO(n) header packet transmits an acknowledgment (ACK(n)) to the compressor. In FIG. 1 it is assumed that T_{dd} is the transmission delay from the decompressor to the compressor, T_{du} is the transmission delay from the compressor to the decompressor, T_{samp} is the time interval between consecutive media samples inserted into the packet, ACK (n) is an acknowledgment transmitted from the decompressor upon receiving an FH packet or an FO(n) header packet, and $SO(_n+T_{dd}+T_{du})/T_{samp})$ is an SO header packet sent by the compressor in response to receipt of the ACK(n)

According to FIG. 1, from time t_0 forward, the compressor continues sending FO header packets through time t_1 until the ACK(n) has been received at the compressor at time t_2. The decompressor, in response to each FO header packet transmitted subsequent to the FO(n) header packet, transmits an ACK to the compressor. At time t_2, once the ACK(n) has been received, the compressor begins transmitting SO header packets to the decompressor. At a time subsequent to time t_2 the decompressor receives the first one of the SO header packets, thereby indicating that the ACK(n) was properly received at the compressor. The decompressor then stops transmitting ACK's to the compressor.

Thus, as is clearly illustrated in FIG. 1 in the time between time t_0 and t_2, FO header packets are still being sent from the compressor, and the decompressor in response to each of these FO header packets sends an ACK, thereby occupying precious bandwidth in the network. Therefore, the conventional technique causes inefficient use of the bandwidth of a network.

SUMMARY OF THE INVENTION

The present invention provides a method and apparatus for eliminating the inefficient use of network bandwidth caused by numerous acknowledgments transmitted by the receiver to the transmitter by providing sparse feedback from the receiver to the transmitter to indicate receipt of packets having headers to be used as reference headers.

The present invention is applicable to a network system where spectral efficiency is a concern. The present invention can also be applied where the compression of the headers of packets which are transmitted in a network system would provide some efficiencies in the use of the bandwidth of the network system. In such a network system, a compression state is established on a link or communication channel between a transmitter (compressor) and receiver (decompressor) so that packets transmitted between the compressor and decompressor on the link or communication channel are sent with compressed headers. The compression state is established by storing information corresponding to information contained in the header of a packet as a context in both the compressor and decompressor, when the header is to be used as a reference header. The compressor and decompressor can, for example, each be separate apparatus provided in the network system or provided as a part of, for example, a router, a host, a terminal or any other such apparatus included in the network system.

In the present invention, a packet having the header to be used as a reference header is transmitted from the transmitter to the receiver. Such a packet can, for example, be a FH packet or a FO packet. According to the present invention, the receiver receives the packet having the reference header and in response provides a feedback to the transmitter indicating receipt of the packet having the reference header. After providing the feedback, the receiver waits a predetermined period of time before providing another feedback in response to another packet having a reference header transmitted by the transmitter.

The predetermined period of time the receiver waits before providing another feedback allows for receipt of information from the transmitter indicating that the transmitter has received the feedback. The predetermined period of time could, for example, correspond to the Round Trip Time of a packet sent from the receiver to the transmitter and back.

The information returned by the transmitter to the receiver in response to the feedback could, for example, be information indicating that the transmitter has altered its behavior. Specifically, the information could, for example, be a packet having a compressed header which corresponds to the reference header. Such a packet can, for example, be a SO header.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be more apparent from the following detailed description, when taken in conjunction with the accompanying drawings, in which:

FIG. 1 illustrates the inefficient use of the bandwidth of a network system according to the conventional technique;

FIG. 2 illustrates an example of a network system architecture in accordance with the present invention;

FIG. 3 illustrates the efficient use of the bandwidth of a network system according to the technique of the present invention; and

FIG. 4 illustrates the efficient use of the bandwidth of a network system according to the technique of the present invention.

DESCRIPTION OF THE INVENTION

The features of present invention are illustrated for example, in FIGS. 2–4. However, it should be understood that the present invention is not limited thereto and can be implemented in other architectures. The present invention is described below as being applicable to a system whereby a compressor and a decompressor is used so as to transmit packets having compressor headers. However, the present invention can be applied to any system where bandwidth requires conservation and a reduction in the number of acknowledgments transmitted from a receiver to a transmitter would improve bandwidth efficiently.

The network system of the present invention as illustrated in FIG. 2 provides a terminal **102** which is connected to an IP network **108**. The terminal **102** can, for example, be a personal computer, telephony apparatus, host, laptop or any other such apparatus which executes processing in accordance with IP/RTP/UDP. Particularly, the terminal **102** can provide packets of voice samples which are formatted according to RTP for transmission over the IP network **108**. In order to accomplish this, the terminal **102** includes an RTP endpoint **104** which identifies terminal **102** (e.g., including IP address, port number, etc) as either a source and destination of RTP packets. While the IP network is provided as an example, other types of packet switched networks can be used in place thereof. Terminal **102** also includes a local timer **103** for generating a time stamp.

An Access Network Infrastructure (ANI) **110** is connected to the IP network **108**. A wireless terminal **130** is coupled via a Radio Frequency (RF) link **140** to the ANI **110**. The RF link **140** includes a up-link **142** which transmits data from the terminal **130** to the ANI **110** and a down-link **144** which transmits data from the ANI 110 to the terminal 130. ANI 110 interfaces one or more wireless or RF terminals including terminal 130 located in different areas of a region to IP network 108. ANI 110 performs functions such as converting between wireline signals provided by the IP network 108 and wireless or RF signals provided by terminals such as terminal 130. Thus, ANI 110 allows RTP packets received from the IP network 108 to be sent over RF link 140 to terminal 130 and allows RTP packets received from, for example, terminal 130 to be sent over the IP network 108 to, for example, terminal 102.

According to the present invention, ANI 110 includes one or more ANI adapters (ANI_AD) such as ANI_AD 112 and ANI_AD 114. Each of the ANI-AD's includes a timer 113 and performs header compression on RTP packets prior to transmitting the packets on the down-link 144 to terminal 130 and performs header decompression on RTP packets after being transmitted on the up-link 142 from terminal 130. The header of each packet includes one or more fields such as a time stamp field. The header of each packet recieved from the IP network 108 is compressed according to RFC 2508 by ANI-AD 112 prior to transmission to terminal 130 on down-link 144. The header of each packet received from the terminal 130 over the up-link 142 is decompressed according to RFC 2508 by ANI_AD 112 before transmission to IP network 108. Therefore, each ANI_AD serve as a compressor and/or a decompressor (compressor/decompressor 115). Thus, the compression/decompression function according to RFC 2508 can be implemented in any of the apparatuses included in the system (e.g., routers, hosts, telephony apparatuses, etc.).

Each ANI_AD interfaces terminals located in a specific area within a region to the IP network 108 and makes use of the timer 113 for implementing a timer-based compression/decompression techniques. ANI_AD 112 also includes a jitter reduction function (JRF) 116 which operates to measure the jitter on packets (or headers) received over the IP network 108 and discard any packets/headers having excessive jitter. Additional ANI's such as ANI 120 are, for example, provided for interfacing other terminals located in other areas of other regions to the IP network 108. ANI 120 similarly includes one or more ANI_AD's such as ANI_AD 122 which includes at least a timer and a JRF as described above.

Terminal 130 includes an RTP endpoint 132 which identifies terminal 130 (e.g., including IP address, port number, etc) as a source and/or destination of RTP packets. Terminal 130 also includes a terminal adapter (TERM_AD) 136 which performs header compression on the headers of packets to be transmitted over the up-link 142 and header decompression on the headers of packets received over the down-link 144. Thus, TERM_AD 136 serves as a compressor or a decompressor (compressor/decompressor 137) similar to ANI_AD. TERM_AD 136 includes a timer 134 for calculating an approximation of a RTP time stamp of a current header and to measure elapsed time between successively received packets.

The configuration illustrated in FIG. 2 is an example of a system in which the present invention is practiced wherein RTP packets are transmitted over a link or communication channel such as the wireless link 140 where bandwidth is at a premium and errors are not uncommon. However, the present invention is not limited to a wireless link but may in fact be applicable to a wide variety of links or communication channels including wireline links. The present invention may, for example, find application in packets which are used for voice over IP network or IP telephony.

In order to illustrate the features of the present invention as it relates to FIG. 2 the following assumptions are made. Data, including, for example, voice, in the form of packets are transmitted from terminal 102 through the IP network 108 to the terminal 130 via ANI 110, ANI_AD 112 and the down-link 144. In order to conserve the bandwidth of the down-link 144, the headers of each of the packets transmitted from the terminal 102 are compressed by the compressor/decompressor 115 which form part of ANI_AD 112. The packets having the compressed headers are transmitted by the compressor/decompressor 115 over the down-link 144 to the terminal 130. The terminal 130 including a compressor/decompressor 137 decompresses the headers of the packets transmitted over the down-link 144 to obtain the original packets. The original packets are then processed by the terminal 130.

In order to initiate a session between the compressor/decompressor 115 and the compressor/decompressor 137 or re-synchronize a compression a state between the compressor/decompressor 115 serving as a compressor and the compressor/decompressor 137 serving as a decompressor, a packet containing a reference header such as Full Header (FH) or a First Order (FO) header is transmitted from the compressor 115 to the decompressor 137. The FH or FO header contains information corresponding to the fields of a header of a packet. Such information is stored as a reference header (context) in the compressor 115 and the decompressor 137. After a session has been initialized or re-synchronization has been performed, all subsequent packets to be transmitted from the compressor 115 to the decompressor 137 are transmitted with compressed headers. The compressed headers can, for example, be of two types. The first type of compressed header is used when the headers of the subsequently transmitted packets can be extrapolated in a linear fashion from the reference header. This type of compressed header is referred to a as a Second Order (SO) header. The FO header is the second type of compressed header. The FO header is used when the headers of the subsequently transmitted packets cannot be extrapolated in a linear fashion. As per the above, both the FH and the FO headers are used as reference headers.

Upon receipt of either an FH packet or an FO packet the decompressor 137 provides a feedback to the compressor 115 indicating receipt of the FH packet or the FO packet. It should be noted that the compressor 115 continues to send FH packets or FO header packets until a feedback from the decompressor 137 has been received indicating proper receipt of the FH packet or the FO header packet. The feedback from the decompressor 137 is sent to the compressor 115 over the up-link 142.

In the conventional technique, an acknowledgment is sent by the decompressor 137 in response to each and every FH packet or the FO header packet which are continually sent by the compressor 115. Thus, the conventional technique inefficiently uses the bandwidth of the up-link 142.

The present invention conserves the bandwidth in the up-link 142 by causing the decompressor 137 to send a feedback in response to a FH packet or a FO header packet and then wait a predetermined period of time before sending another feedback in response to FH packets or FO header packets which are continually being transmitted by the compressor 115. Thus, the decompressor 137 transmits one feedback over the up-link 142 in response to the FH packet or the FO header packet and waits for an indication from the compressor 115 that the feedback has been received. If after the predetermined period of time has expired, an indication that the compressor 115 received the feedback has not been received then the decompressor **137** sends another feedback in response to a subsequently transmitted FH packet or FO header packet.

The predetermined period of time during which the decompressor **137** waits before sending another feedback in response to a subsequently transmitted FH packet or FO header packet, allows time for the feedback to traverse the up-link **142** to the compressor **115** and time for the indication of receipt of the feedback by the compressor **115** to traverse the down-link **144** back to the decompressor. The predetermined period of time could, for example, correspond to the Round-Rrip-Time (RTT) it would take for a packet transmitted from the decompressor **137** to the compressor **115** over the up-link **142** and for the packet to be returned back to the decompressor **137** over the down-link **144**. The information from the compressor **115** indicating that the compressor has received the feedback could be information indicating that the compressor **115** has altered its behavior by, for example, sending SO header packets.

The basic feature of the invention is that it is more likely than not that the feedback transmitted by the decompressor **137** to the compressor **115** will properly traverse the link between the decompressor **137** and compressor **115** so as to cause the compressor **115** to alter its behavior. Since it is more likely than not for the feedback to properly traverse the link between the decompresssor **137** and the compressor **115**, then it would be on rare occasions that the feedback is not received by the compressor **115**. When such occurs, the decompressor re-transmits another feedback in response to a subsequently transmitted FH packet or the FO header packet.

The technique of the present invention is particularly illustrated in FIG. **3**. In FIG. **3**, similar to FIG. **1**, where T_{dd} is the transmission delay from the decompresssor **137** to the compressor **115**, T_{du} is the transmission delay from the compressor to the decompresssor **137**, T_{samp} is the time interval between consecutive media samples, HDR(n) is a packet having a header that can be used as a reference header (FH or FO header) transmitted from the compressor **115** with a sequence number n, FB(n) is a feedback sent from the decompresssor **137** to the compressor **115** upon receiving a packet with the header HDR(n), T_{round} is the round-trip delay for a packet to traverse the link between the decompressor **137** and compressor **115** and back, where T_{round} equals $T_{dd}+T_{du}$, CH and is a packet having a compressed header that may be of the FO less optimal state or the SO more optimal state.

As illustrated in FIG. **3**, at time to the compressor **115** transmits a packet having a header HDR(n) to the decompresssor **137**. At time t_1 the decompressor **137** detects the packet having the header HDR(n) and in response transmits the feedback FB(n) to the compressor **115**. In the meantime, the compressor **115** continues to transmit packets having the header HDR through time t_1. The decompresssor **137** after transmitting the feedback FB(n) to the compressor **115** waits a predetermined period of time corresponding to, for example, T_{round}. In other words, the decompresssor **137** does not send another feedback to any further packets having the header HDR until the time T_{round} has elapsed.

Once the compressor **115** receives the feedback FB(n) at time t_2, the compressor **115** alters its behavior and begins sending packets having compressed headers $CH(n+T_{dd}+T_{du})/T_{samp})$ The decompresssor **137** receives the $CH(n+T_{dd}+T_{du})/T_{samp})$ packets having the compressed headers from the compressor **115**, thereby indicating that the feedback FB(n) was properly received by the compressor **115**.

FIG. 4 illustrates a situation according to the present invention where the feedback FB(n) was lost due to, for example, a link layer error. In such a situation, the decompresssor **137** waits until the predetermined period of time has elapsed and then determines that no information or indication has been transmitted from the compressor **115** that the feedback FB(n) has been received. The decompresssor **137** then transmits another feedback FB to the compressor **115** in response to a subsequently transmitted packet having a header HDR and waits for another predetermined period of time. Of course, if the compressor **115** does not transmit an indication that the re-transmitted feedback FB has been received, then at the very least the compressor **115** and decompressor **137** would remain in a less optimal state. The less optimal state is where packets having headers which are to be used as reference headers (FH or FO) are transmitted.

Even where an indication of receipt of the feedback has not been received, the present invention offers advantages being that there still is a reduced number of feedback transmitted from the decompressor **137** to the compressor **115** relative to the number of acknowledgements that would be transmitted according to the conventional technique. According to the present invention, subsequent feedbacks are only sent after the predetermined period of time has elapsed. Thus, the present invention provides for a sparse number of feedbacks to be transmitted from the decompressor **137** to the compressor **115**.

According to the above, the present invention provides a method and apparatus for eliminating the inefficient use of network bandwidth caused by numerous acknowledgments transmitted by a receiver to a transmitter. The present invention accomplishes this by providing sparse feedback from the receiver to the transmitter to indicate receipt of packets having headers to be used as reference header. More particularly, the present invention upon receipt in the receiver of a packet having a reference header, provides a feedback to the transmitter indicating receipt of the packet having the reference header. The receiver then waits a predetermined period of time before providing another feedback in response to another packet having a reference header. This period of time allows for the feedback to traverse the link between the receiver and information indicating that the transmitter received the feedback to traverse the link between the transmitter an the receiver. Thus, the present invention makes for efficient use of the bandwidth of a network by providing a sparse number of feedbacks to a transmitter acknowledging receipt of a packet having, for example, a reference header.

While the present invention has been described in detail and pictorially in the accompanying drawings it is not limited to such details since many changes and modifications recognizable to those of ordinary skill in the art may be made to the invention without departing from the spirit and the scope thereof.

We claim:

1. In a system having a transmitter which transmits a plurality of packets to a receiver, each of the packets containing a header, a method of providing sparse feedback from the receiver to the transmitter indicating receipt of a packet having a header to be used as reference header, comprising:

transmitting from the transmitter to the receiver a packet having a header to be used as a reference header;

receiving in the receiver said packet having the reference header and in response providing feedback from the receiver to the transmitter indicating receipt of said packet having the reference header; and

waiting a predetermined period of time before providing another feedback in response to another packet having the reference header to permit receipt of information in the receiver indicating that the transmitter received said feedback,

wherein said information received in the receiver includes information transmitted by the transmitter indicating that the transmitter has altered its behavior based on receipt of said feedback.

2. The method according to claim **1**, wherein said information transmitted by the transmitter is a packet having a compressed header which is related to said reference header.

3. The method according to claim **2**, wherein said reference header is a full header (FH) or a first order (FO) header.

4. The method according to claim **3**, wherein said information transmitted by the transmitter to the receiver indicating its altered behavior is one of a FO header when said reference header is a FH, and a second order (SO) header when said reference header is said FH or said FO header.

5. The method according to claim **4**, wherein the altered behavior of the transmitter includes one of switching from a FH state to a FO state and initiating transmission of packets each having a FO header, and switching from either a FH state or an FO state to a SO state and initiating transmission of packets each having a SO header.

6. The method according to claim **1**, wherein said predetermined period of time is a round trip time (RTT) representing a time it takes for a packet to be transmitted from the receiver to the transmitter and for the packet to be returned by the transmitter back to the receiver.

7. The method according to claim **2**, wherein said predetermined period of time is a round trip time (RTT) representing a time it takes for a packet to be transmitted from the receiver to the transmitter and for the packet to be returned by the transmitter back to the receiver.

8. The method according to claim **3**, wherein said predetermined period of time is a round trip time (RTT) representing a time it takes for a packet to be transmitted from the receiver to the transmitter and for the packet to be returned by the transmitter back to the receiver.

9. The method according to claim **4**, wherein said predetermined period of time is a round trip time (RTT) representing a time it takes for a packet to be transmitted from the receiver to the transmitter and for the packet to be returned by the transmitter back to the receiver.

10. The method according to claim **5**, wherein said predetermined period of time is a round trip time (RTT) representing a time it takes for a packet to be transmitted from the receiver to the transmitter and for the packet to be returned by the transmitter back to the receiver.

* * * * *

US005307413A

United States Patent [19]

Denzer

[11] Patent Number: 5,307,413

[45] Date of Patent: Apr. 26, 1994

[54] METHOD AND APPARATUS FOR ADDING DATA COMPRESSION AND OTHER SERVICES IN A COMPUTER NETWORK

[75] Inventor: Philip C. Denzer, Wellesley, Mass.

[73] Assignee: Process Software Corporation, Framingham, Mass.

[21] Appl. No.: 733,104

[22] Filed: Jul. 19, 1991

[51] Int. Cl.⁵ H04B 1/66; H04L 9/00

[52] U.S. Cl. 380/49; 380/9; 370/109; 375/122

[58] Field of Search 380/4, 9, 25, 49, 50; 395/21, 24, 25; 364/240.8, 242.94–242.96, 274.9, 276.6, 940.61–940.62, 940.81, 972.4; 375/122; 340/825.8, 825.5; 381/34, 35; 370/109

[56] References Cited

U.S. PATENT DOCUMENTS

4,533,948	8/1985	McNamara et al.	340/825.5 X
4,691,314	9/1987	Bergins et al.	370/94
4,748,638	5/1988	Friedman et al.	375/8
4,833,468	5/1989	Larson et al.	340/825.8
4,972,473	11/1990	Ejiri et al.	380/9 X

OTHER PUBLICATIONS

Strass, Hermann, "D-A-T-A Compression," *DEC Professional*, Feb. 1991, pp. 58–62.

Turner, Steven E., "Small Is Beautiful: How V.42bis Cuts Costs," *Data Communications*, Dec. 1990, pp. 101–104.

Welch, Terry A., "A Technique for High-Performance Data Compression," *Computer* Magazine, vol. 17 No. 6 (Jun. 1984), pp. 8–19.

"Sotfware Product Description FCX Version 3.2," brochure of Innovative Computer Systems, Inc., (ICS), 1930 E. Marlton Pike, Cherry Hill, N.J. 08003, pp. 1–4, no date.

"FCX-pcFCX-File Compression for VAX/VMX and MS-DOS Systems," brochure of Innovative Computer Systems, Inc. (ICS, Inc.), 1930 E. Marlton Pike, Cherry Hill, N.J. 08003, (4 pages), no date.

F. Jay, *IEEE Standard Dictionary of Electrical and Electronics Terms*, (ANSI/IEEE Std. 100–1984); (IEEE, New York, 1984, p. 695, "Protocol").

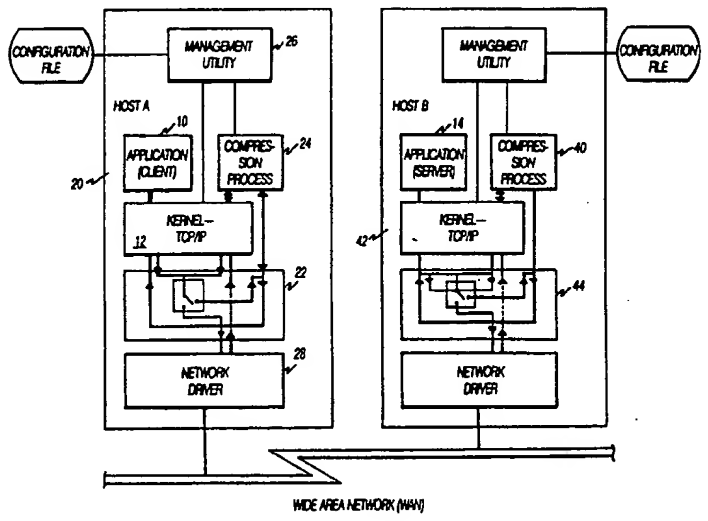
Primary Examiner—Bernarr E. Gregory

Attorney, Agent, or Firm—Weingarten, Schurgin, Gagnebin & Hayes

[57] ABSTRACT

A connection specific compression system is selectively implemented in connections having the greatest data redundancy and utilizes modularity in implementing data compression in a layered network communication system. A data compression facility is interfaced in the layered system and intercepts data at a protocol layer prior to the data being packetized for transmission. A system acting as a compression host comprises a data packet switch driver which intercepts application data packets passing over layered network interfaces and routes selected client application data packets to an associated local compression process which has an integral network protocol and which compresses the data stream in accordance with a selected compression algorithm. The compressed data passes through the system network protocol and the packet switch driver subsequently sends the compressed data back into the communications stream through a network driver. The compressed data passes across the network communication channel and is received by a decompression host having peer compression/decompression capabilities. The peer compression process decompresses the received data and sends it, via a second/decompression host resident packet switch driver, as though received from the network, into the decompression host system network protocol for connection with an application running on the second host.

19 Claims, 9 Drawing Sheets



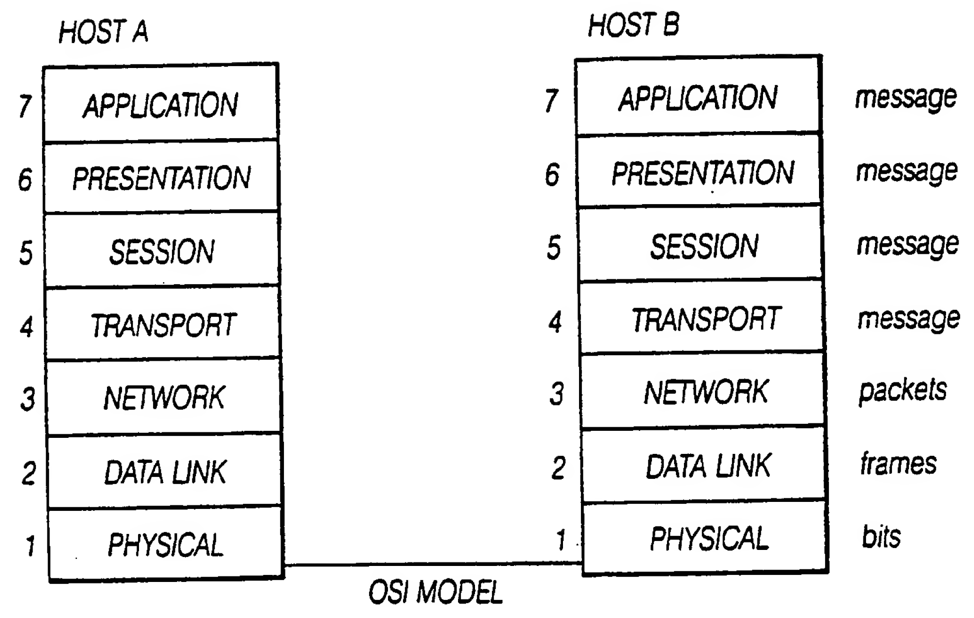


Fig. 1
(PRIOR ART)

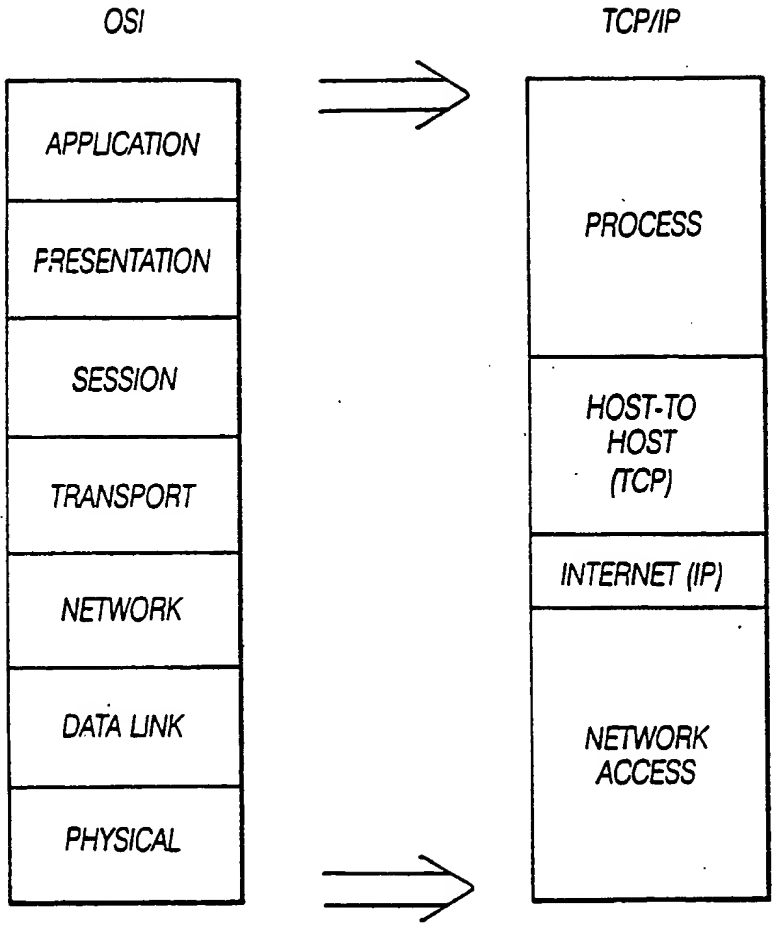
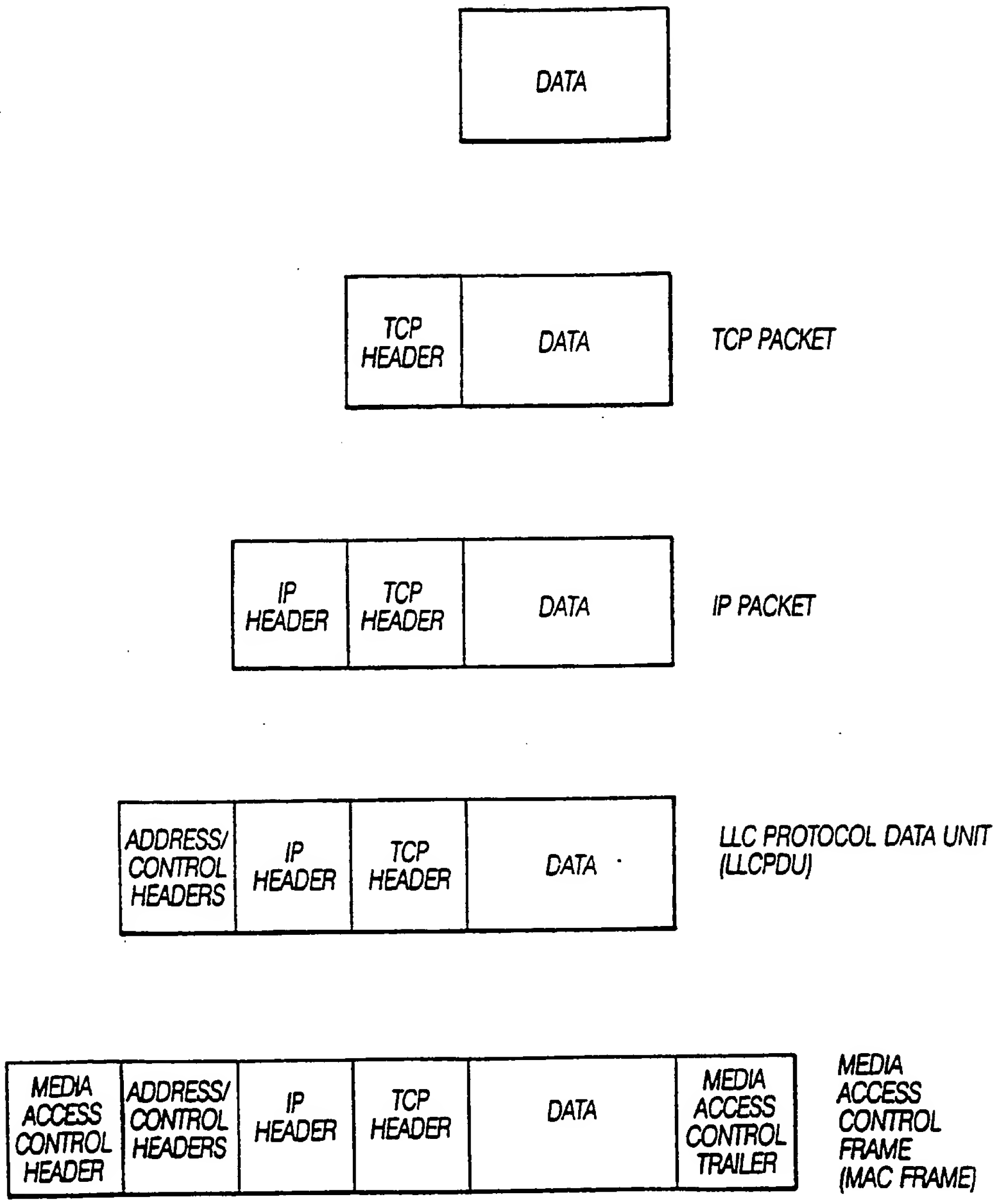


Fig. 1a
(PRIOR ART)



TCP/IP PACKETIZATION

Fig. 2

(PRIOR ART)

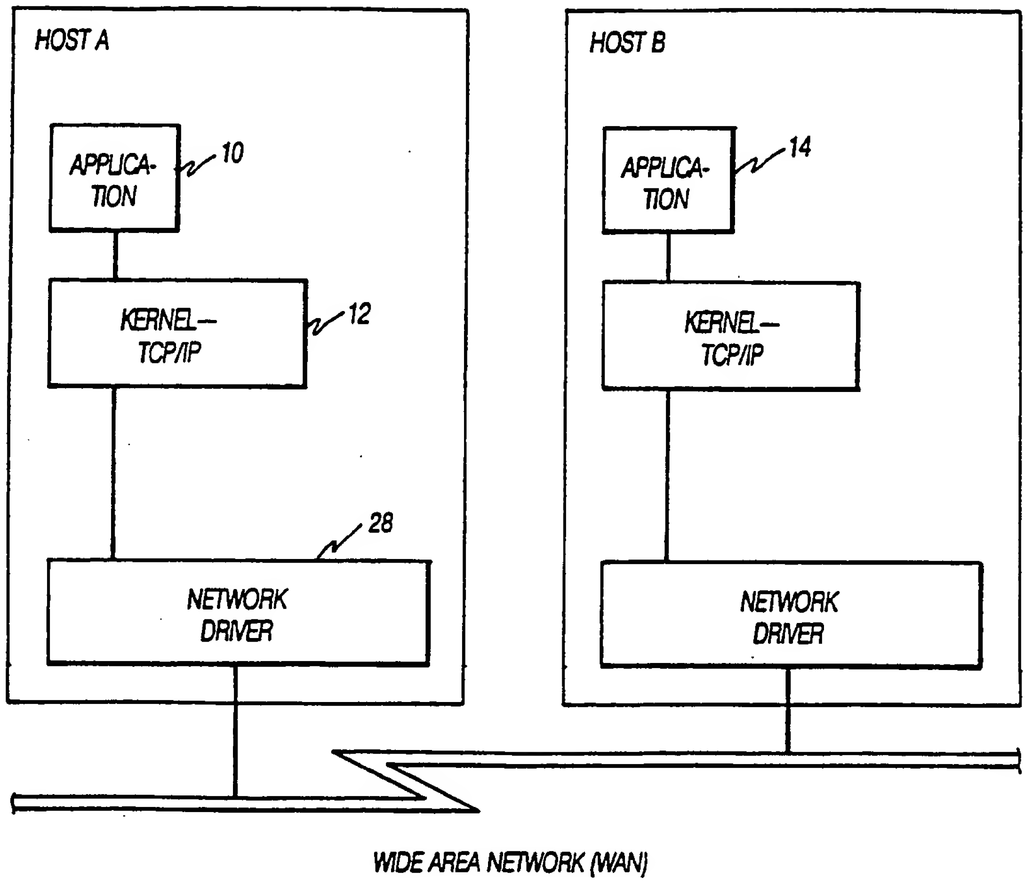


Fig. 3
(PRIOR ART)

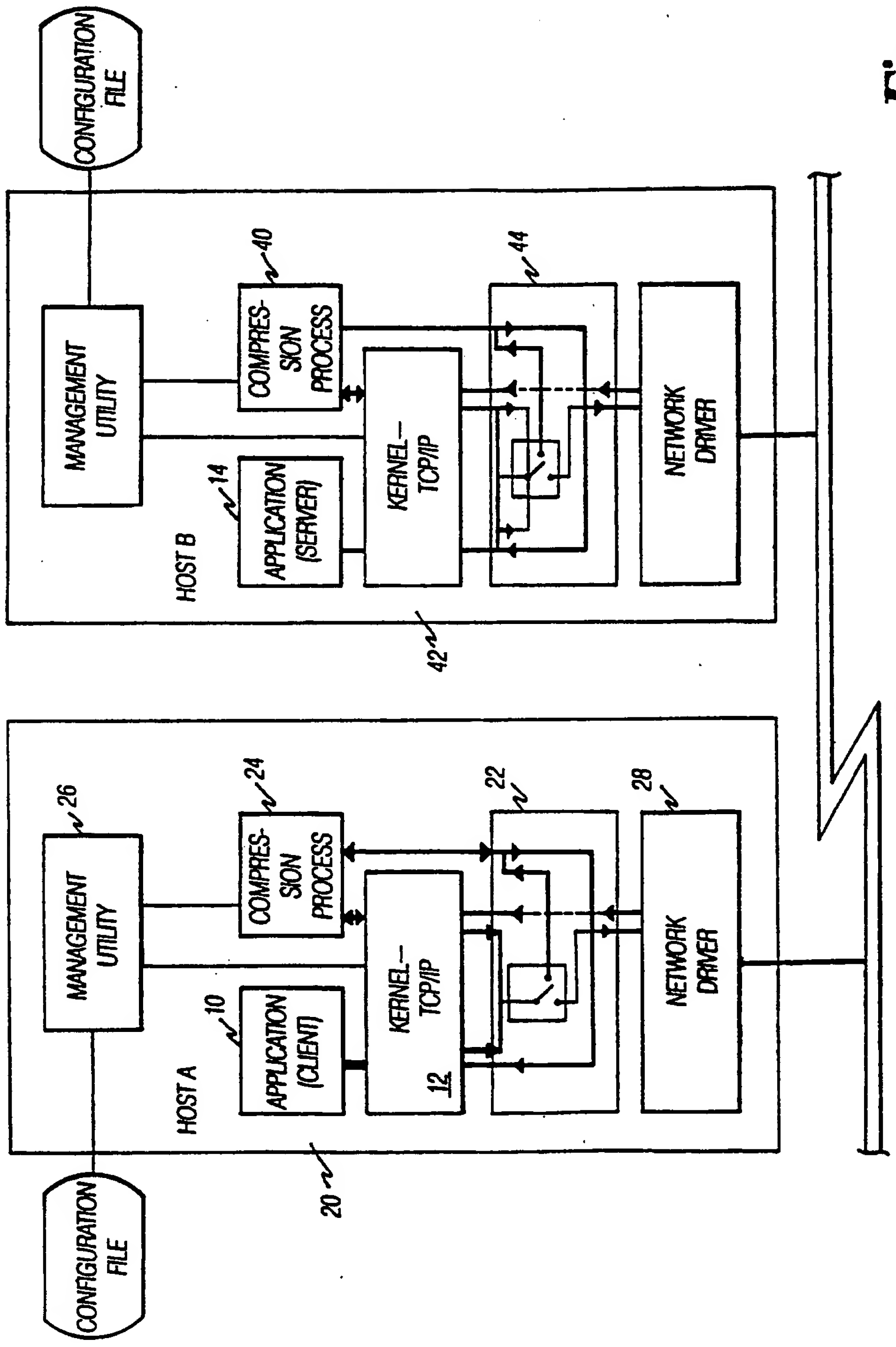

CONFIGURATION FILE
MANAGEMENT UTILITY
COMPRES-SION PROCESS
40
44
APPLICATION (SERVER)
14
KERNEL—TCP/IP
NETWORK DRIVER
HOST B
42
WIDE AREA NETWORK (WAN)
26
24
22
28
MANAGEMENT UTILITY
COMPRES-SION PROCESS
APPLICATION (CLIENT)
10
KERNEL—TCP/IP
12
NETWORK DRIVER
HOST A
20
CONFIGURATION FILE

Fig. 4

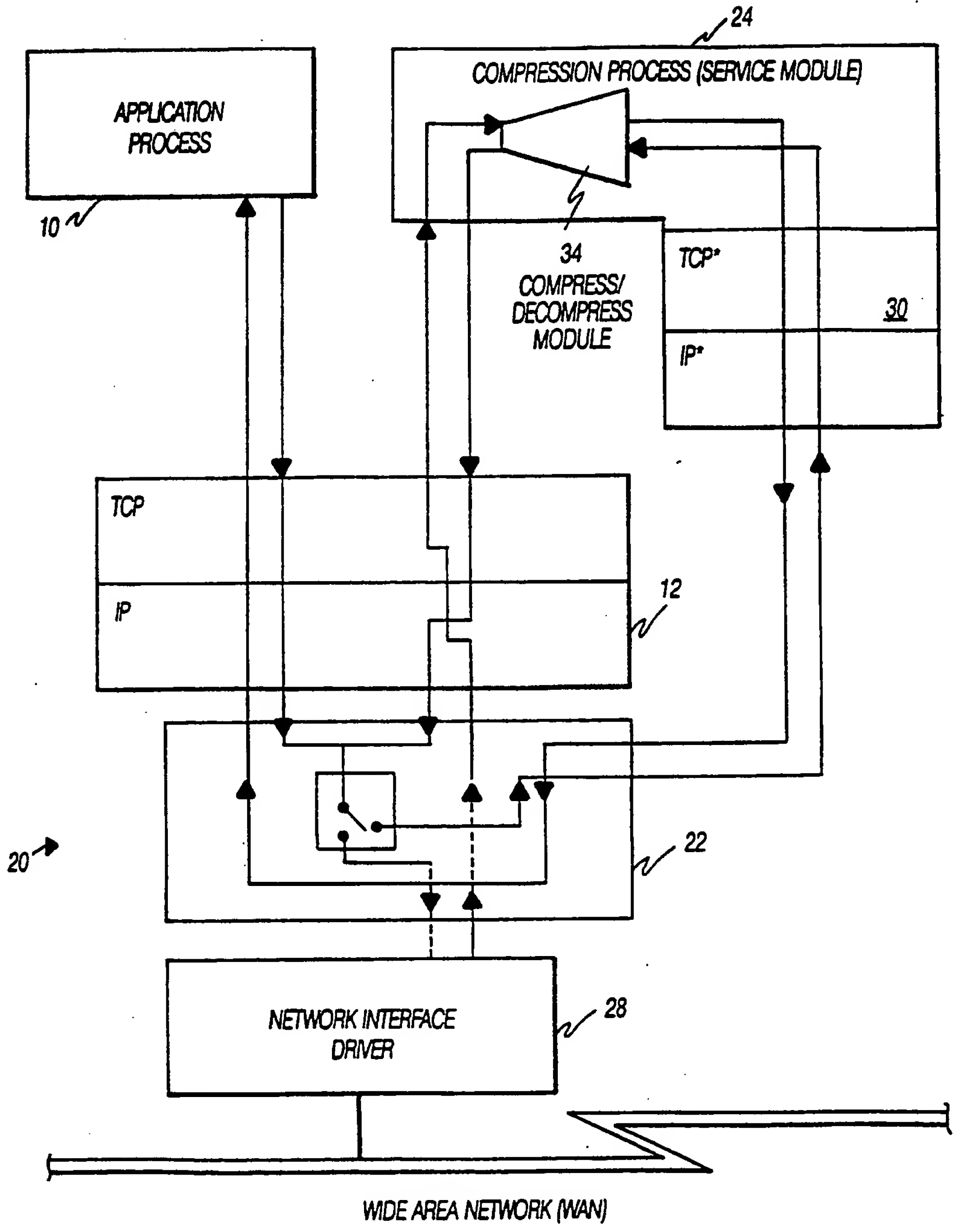

24
COMPRESSION PROCESS (SERVICE MODULE)
APPLICATION PROCESS
10
34
COMPRESS/ DECOMPRESS MODULE
TCP*
30
IP*
TCP
IP
12
20
22
NETWORK INTERFACE DRIVER
28
WIDE AREA NETWORK (WAN)

Fig. 4a

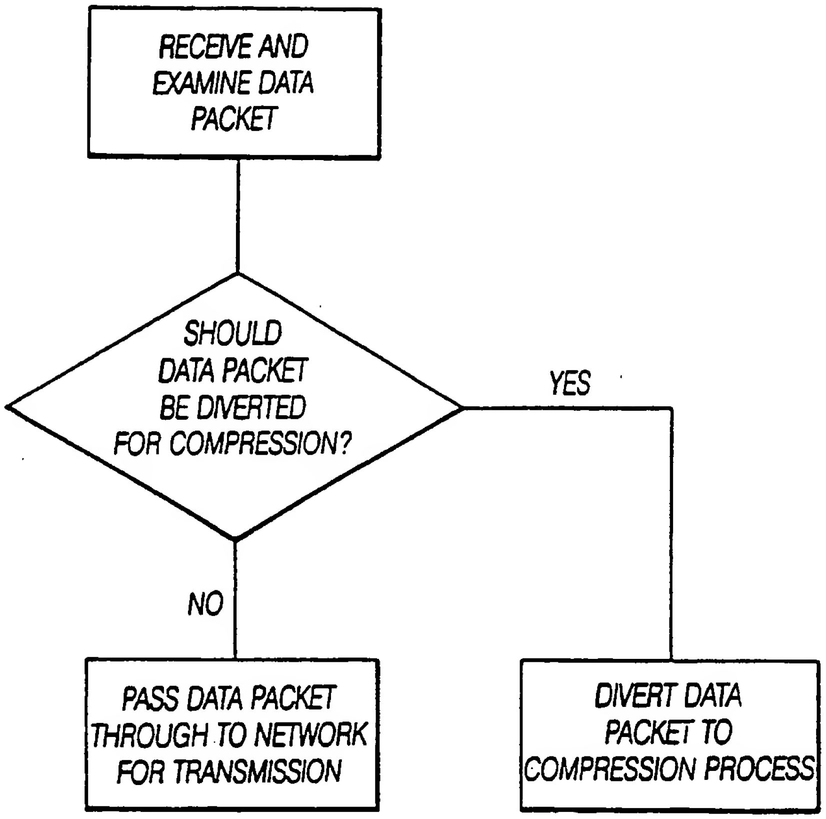

RECEIVE AND EXAMINE DATA PACKET
SHOULD DATA PACKET BE DIVERTED FOR COMPRESSION?
YES
NO
PASS DATA PACKET THROUGH TO NETWORK FOR TRANSMISSION
DIVERT DATA PACKET TO COMPRESSION PROCESS

Fig. 5

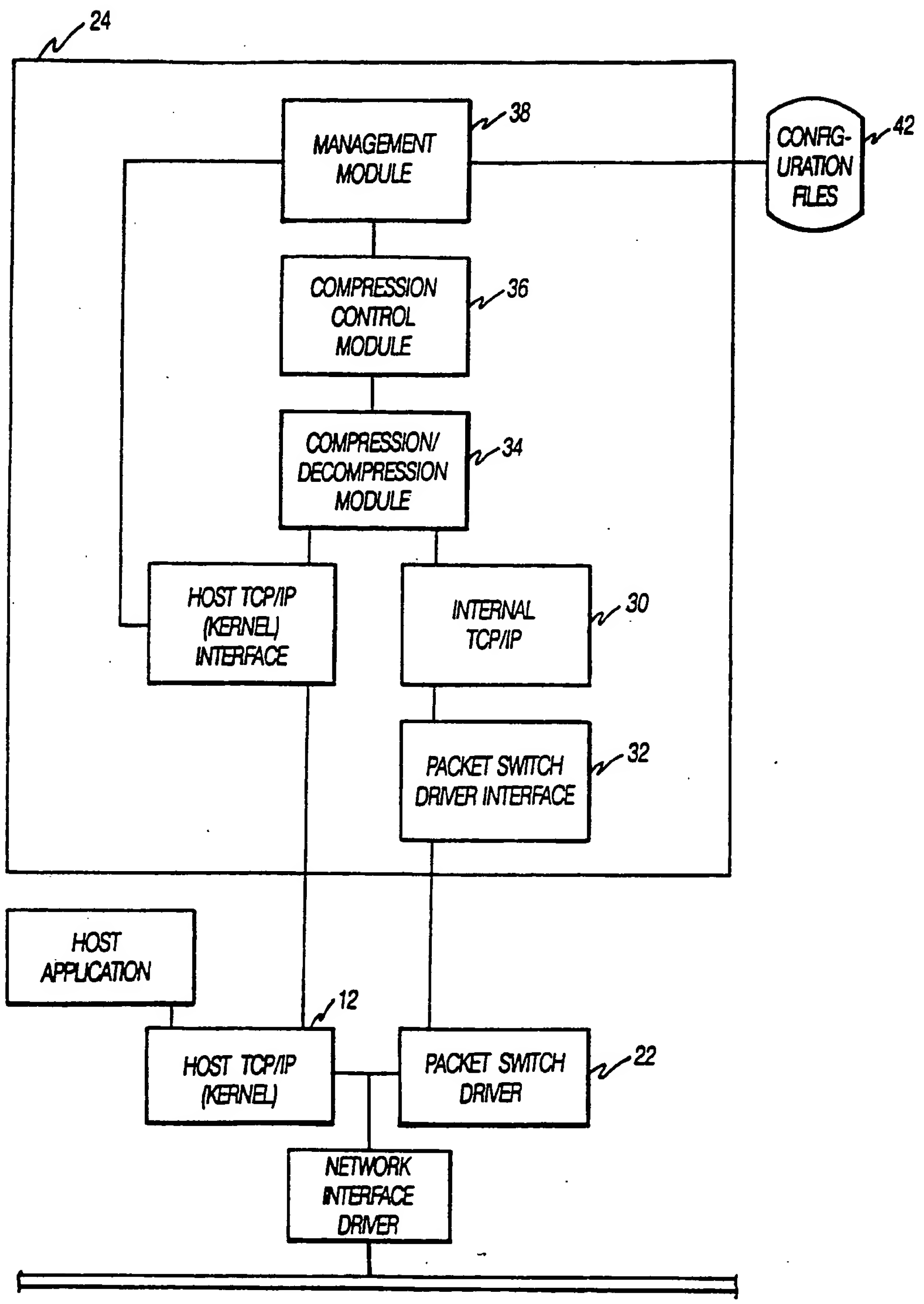

24
MANAGEMENT MODULE
38
CONFIG-URATION FILES
42
COMPRESSION CONTROL MODULE
36
COMPRESSION/ DECOMPRESSION MODULE
34
HOST TCP/IP (KERNEL) INTERFACE
INTERNAL TCP/IP
30
PACKET SWITCH DRIVER INTERFACE
32
HOST APPLICATION
12
HOST TCP/IP (KERNEL)
PACKET SWITCH DRIVER
22
NETWORK INTERFACE DRIVER

Fig. 6

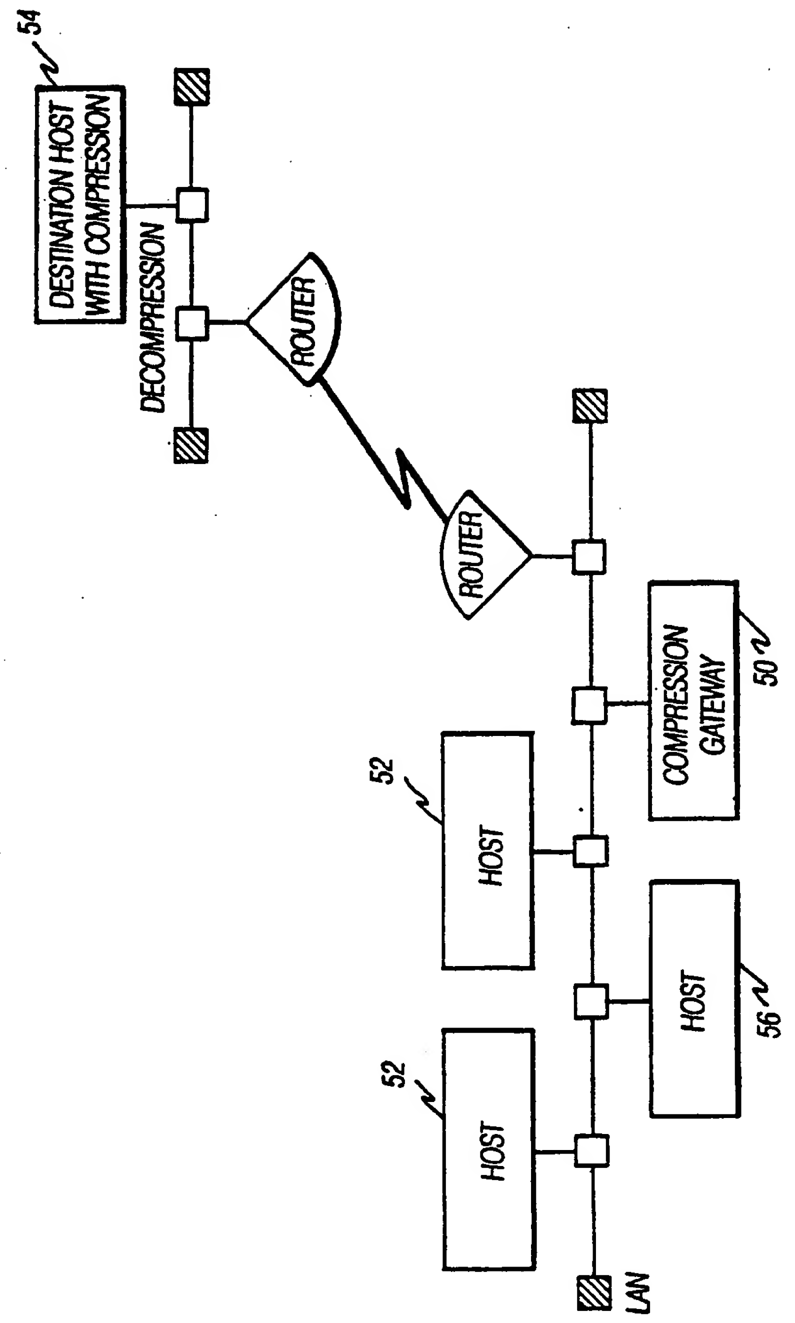

54
DESTINATION HOST WITH COMPRESSION
DECOMPRESSION
ROUTER
ROUTER
COMPRESSION GATEWAY
50
52
HOST
52
HOST
HOST
56
LAN

Fig. 7

METHOD AND APPARATUS FOR ADDING DATA COMPRESSION AND OTHER SERVICES IN A COMPUTER NETWORK

FIELD OF THE INVENTION

The present invention relates to data compression and in particular to data compression in the context of networked computers.

BACKGROUND OF THE INVENTION

Systems are known which provide for connectivity in and among networks of computerized equipment. Such systems, ideally, must accommodate a proliferation of different network interfaces and hardware, as no single "standard" is adopted universally.

In order to permit computer systems to communicate, regardless of connection method or vendor-specific hardware implementation, or to permit different networks to communicate or be "internetworked", modularized/layered solutions have been proposed for the problems associated with interconnectivity. Layering divides the task of interconnection and communication into pieces (layers), wherein each layer solves a piece of the problem or provides a particular function and is interfaced to adjacent layers. Each of the layers is responsible for providing a service to ensure that the communication is properly effected. Examples of some services provided by the various layers are error detection, error recovery and routing among many communication paths. All the layers in conjunction present the overall communication solution. Experience has shown that modularizing in layers with well defined functional interfaces, divides and effectively reduces the complexity of the connectivity problem and leads to a more flexible and extensible solution.

A model for describing the layers in a network has been posited by the International Standards Organization (ISO). The ISO open systems interconnection (OSI) model is a seven-layer model, illustrated in FIG. 1, which provides a standard for describing a network and facilitating computer communications. OSI and other layered computer network communications standards are discussed in detail in *Unix Network Programming* by W. Richard Stevens, and *Handbook of Computer-Communication Standards* by William Stallings, which are incorporated herein by reference. The OSI model defines the layers and units of information that pass along a network. As illustrated, data from an application or process running on a first host (HOST A) moves down the model network layers to a Physical layer. The Physical layer defines the physical connection which transmits raw bits across a communication channel to another host (HOST B) and up corresponding layers to a process running thereon. OSI, while defining a model or framework in which standards and protocols can be developed at each layer, allows for a flexible approach for implementation of the model.

Layered protocols and interfaces therebetween have been defined, which provide specifications for communication between a process or program being executed on one computer's operating system and another process running on another computer. Transmission Control Protocol/Internet Protocol (TCP/IP) are two protocols that are part of a protocol suite or family of protocols layered and designed to connect computer systems that use different operating systems and network technologies. TCP/IP, which provides a common set of

protocols for invocation on dissimilar interconnected systems, is illustrated and mapped in FIG. 1*a* to analogous layers of the OSI model.

TCP/IP is a four layer protocol suite which facilitates the interconnection of two or more computer systems on the same or different networks and in certain networks, such as the Internet, is a requirement for interoperability. The four layers, comprise two independent protocols: TCP which can be used to access applications on other systems within a single network; and IP which permits identification of source and destination addresses for communication between systems on different networks.

As illustrated in FIG. 2, application or process data communicated via TCP/IP is "packetized" as it passes down layers through the protocol suite. The original process data first has an information block called a TCP Header prefatorily appended thereto in a TCP layer, to form a TCP packet. The TCP Header contains information to assure that the data travels from point to point reliably without picking up errors or getting lost. An IP layer repacketizes the TCP packet into an IP packet, by adding an IP Header which contains information needed to get the packet to a destination node. The IP packet is further packetized, such as in ANSI/IEEE 802 local area network protocol, with an additional Logical Link Control (LLC) address header and a control header at an LLC layer, to form an LLC Protocol Data Unit (LLCPDU). The LLCPDU is "framed" for transmission by addition of a Media Access Control Header and Trailer, to form a MAC Frame for communication between two TCP/IP facilities.

It is apparent that a considerable amount of "baggage", in the form of headers and trailer, is added to data which is transmitted between facilities using a layered protocol suite, such as TCP/IP and other OSI modelled families. Many additional bits are added at the various layers and must be processed for ultimate transmission across a communication channel at the physical layer. At its destination, the transmitted frame must be unpacketized according to embedded instructions and passed upward through the layered protocols to its receiving application or process.

Aside from the significant processing overhead associated with packetizing data for network and internetwork transmission, data itself may be redundant, such that real costs are associated with putting data and all its protocol suite baggage across the communication channel.

Where the communication channel is integral to a wide area network (WAN) the impact of redundant data and protocol suite baggage on transmission costs can be measured in dollars. WAN links, such as dial-up phone lines, typically have relatively low, fixed upper limit data transmission rates and are billed based on connect time, packet count or bandwidth use. Thus, the extensive packetization and resultant bits that must be transmitted increase connect time and dollar cost. Additionally, the numerous appended bits significantly increase the possibility of transmission errors on less reliable lines. Where WAN links, such as analog leased lines, and Dataphone Digital Services (DDS) are used, monthly or annual subscription fees recur and to a great extent are consumed by traffic comprising the transmission of redundant bits and protocol attributable header and trailer information.

Compression schemes and apparatus are known which reduce the quantity of bits that must be transmitted across the communication channel. However, data compressing modems, such as disclosed in U.S. Pat. No. 4,748,639, compress data streams after packetization for transmission over the communication channel. Significant additional hardware is required, in the form of a compression modem and a decompression modem. The additional hardware, which translates into significantly increased system cost, requires frequency tables to recodify and decodify or decompress characters in a data stream in accordance with a compressed character code. Such a compression scheme is not host resident, does not take advantage of the modular/layered structure of the network communication system and lacks the flexibility to reduce the number of packets and headers generated in the transmission of a particular data stream. Further, because compression modems service multiple connections and data that may come from one or more systems, redundancy of data tends to be random. It is appreciated that randomly redundant data is more difficult to compress.

Network bridge products are available that also have data compression facilities. However, much like compression modems, bridges do not provide host resident end-to-end compression facilities. Bridges also do not have the capability to reduce packet count. Like compression modems, bridge products are not connection specific and must process data from various connections which tend to provide data that is randomly redundant. Thus data compression with bridge products may not achieve optimal compression ratios.

Host resident compression is available with application programs that incorporate compression algorithms. However, such programs typically retrieve a file from a mass storage device such as disk storage, process the file and return it to the disk. The processed file must then be retrieved from the disk to be shipped through the network layers for transmission. In addition to the significant additional overhead required for the disk storage and retrieval, such compression applications require that process data be effectively compiled as a compressed data file and recompiled or decompressed at a destination facility. Such compression and decompression significantly delays the availability of data subjected to this process of data transmission. Such decompression programs, although host resident, are typically not integrated with particular applications and must be invoked so that they are not transparent to the user.

SUMMARY OF THE INVENTION

The present invention is a connection specific compression system which is selectively implemented in connections having the greatest data redundancy and takes advantage of modularity in implementing data compression in a layered network communication system. A data compression facility is interfaced in the layered system and intercepts data at a protocol layer prior to the data being packetized for transmission. Therefore, fewer packets need to be created and fewer headers and trailers need to be generated to transmit the compressed data.

According to the invention, a system acting as a compression host comprises a data packet switch driver which intercepts application data packets passing through layered network interfaces, prior to further packetization. When the system is acting as a compression host and transmitting data to an application on a

second/decompression host, the data packet switch driver establishes a connection between a client application on the compression host and a system network protocol kernel interface (commonly called a socket) and routes selected client application data packets to an associated local compression process which has an integral network protocol. The data packets selected for compression are delivered to the compression process which opens a second connection to the system network protocol interface and compresses the data stream in accordance with a selected compression algorithm. The compressed data passes through the system network protocol and the packet switch driver subsequently sends the compressed data back into the communications stream through a network driver. The compressed data passes across the network communication channel and is received by the decompression host having compression/decompression capabilities according to the invention.

The decompression host receives the compressed data through its network interface and delivers it to a peer compression process having its own complimentary integral network protocol. The peer compression process decompresses the received data and sends it, via a second/decompression host resident packet switch driver, as though received from the network, into the decompression host system network protocol for connection with an application running on the second host.

In another embodiment, a host system having data compression according to the invention is used as a gateway to process transmissions of systems not having compression according to the invention, which are destined for another host having compression according to the invention.

Features of the invention include end-to-end enhanced data transmission having data compression that is transparent to the user effected between at least two hosts. Reduction in data and number of packets transmitted across the communication channel improves Wide Area Network performance and response time while traffic is reduced to effectively increase WAN bandwidth. Compression according to the invention can be used with various network interfaces supported by the host because it operates independently of the network physical layer. Bridges and routers on the network will not affect, or be affected by, transmission of data serviced according to the invention.

A compression management facility permits system manager invocation and definition of a configuration file of paths or connections in the network, which are to be compressed. Parameters can be set and displayed relating to data to be selected for compression. Statistics can be compiled and logged, to track: number of packets sent and received; number of connections serviced; number of connections open; compression ratio; and compression paths and interfaces in use.

DESCRIPTION OF THE DRAWING

The invention will be more fully understood from the following detailed description taken in conjunction with the accompanying drawing in which:

FIG. 1 is a diagrammatic view of an Open Systems Interconnection (OSI) model according to the prior art;

FIG. 1*a* is a diagrammatic view of the OSI model of FIG. 1 compared to a Transmission Control Protocol-/Internet Protocol (TCP/IP) model according to the prior art;

FIG. 2 is a diagrammatic view of data packetized in accordance with the TCP/IP model of FIG. 1*a;*

FIG. 3 is a block diagram of an illustrative TCP/IP connection from a first host to a second host across a Wide Area Network, according to the prior art, without data compression;

FIG. 4 is a block diagram of a two host end-to-end connection having compression according to the invention in a TCP/IP implementation;

FIG. 4*a* is a block diagram of a host having compression according to the invention in a TCP/IP implementation;

FIG. 5 is a flow chart of functions of a packet switch driver of FIG. 4*a;*

FIG. 6 is a block diagram of a compression process of FIG. 4*a;* and

FIG. 7 is a block diagram of a system having compression according to the invention implemented as a gateway.

DETAILED DESCRIPTION

A host-to-host interconnection of two computers having ~~TCP/IP~~ network capabilities for communicating across a Wide Area Network, as illustrated in FIG. 3, provides a suitable environment for discussion of the present invention. Without compression, such an interconnection comprises an application 10 running under an operating system on a first system, Host A, which transfers data to a socket or interface of a TCP/IP protocol stack 12 comprising two layers, TCP and IP, of the four layers of a TCP/IP protocol suite for packetizing the data for transmission, as discussed hereinbefore. Fully TCP/IP packetized "datagrams" are delivered to a network interface driver, which effects further packetization, framing and transmission of the data on the WAN medium. A network interface driver at the receiving host, Host B, strips and passes the datagrams, including the data, to a Host B TCP/IP kernel where the data is de-packetized and a connection is accepted and made to an application or process running thereon which reads the data.

Referring now to FIG. 4, a system 20 ~~having TCP/IP capabilities and data compression according to the invention comprises a first~~ host, Host A, having a packet switch driver 22 installed between the host TCP/IP kernel 12 which receives data from the application or process 10, and a network interface driver 28 that effects a lower layer of packetization, framing and transmission of the data on the WAN. The packet switch driver 22 communicates with a local compression process 24, which is itself in communication with the host TCP/IP kernel 12 and functions as described hereinafter. A management utility 26 oversees and facilitates configuration of compression within the system 20, according to the invention.

When the application 10, such as one running on Host A, seeks to communicate with a remote application 14, such as one running on Host B, a request and data are passed to the host TCP/IP layers 12 running in an operating system kernel. The packet switch driver 22, as illustrated in FIGS. 4, 4*a* and 5, receives all data travelling to the network interfaces from an application. ~~The packet switch driver 22 examines all such data and intercepts selected data for routing to the local compression process 24, according to predetermined criteria.~~

~~Data is selected for compression based upon operating parameters~~, such as network interface ~~addresses~~ and paths, established via the management utility 26 and maintained by the compression process 24 discussed hereinafter. ~~The packet switch driver 22 compares the data packet address appended in the TCP/IP layers and determines if the data packet should be routed for compression. If the packet switch driver 22 determines that the data packet should not be compressed, that data packet is passed through to the network driver 28 to continue untouched in the communication stream. The selected data packets, in accordance with addresses listed in a configuration file established via the management utility 26, are switched to the local compression process 24.~~

Packets intercepted by the packet switch driver 22 for compression are passed through a compression process-packet switch driver interface 32 (FIG. 6) and received by internal TCP/IP layers 30 that are functionally interrelated with the compression process 24, and not part of the host TCP/IP 12 running under the operating system. The compression process 24, illustrated in FIGS. 4*a* and 6, functioning with the internal TCP/IP layers 30, performs TCP/IP input processing which assures the reliability of the data stream in the packets received from the packet switch driver 22.

A compression/decompression module 34 represents a service module which is modularly integrated with the internal TCP/IP layers 30 and communicates with the host TCP/IP 12 for transmitting data on the network. The compression/decompression module/34 comprises a general purpose compression/decompression algorithm, ~~which is interfaced to permit different compression methods to be easily integrated~~. A compression/decompression module 34, in one example, is an implementation of a ~~Lempel-Ziv algorithm for sequential data compression~~, which is deselectable or modularly removable to ~~permit selection of an alternative algorithm known in~~ the art, such as an LZW compression implementation or Huffman encoding. Data received by the compression process from the packet switch driver 22 is compressed for transmission by the compression/decompression module 34 ~~in accordance with the selected algorithm~~.

A compression control module 36 in the compression process 24, in conjunction with a management module 38 that is a compression process resident portion of the management utility 26, effect two phases in communication of compressed data from the compression process 24 to a remote compression process 40 (FIG. 4), within remote Host B.

A first phase in communication of compressed data between compression processes involves the negotiation of connections. If it is determined by the switch 22 that a request to connect to a remote application requires invocation of data compression, the local compression process 24 that receives the diverted data determines if a connection needs to be made to the remote compression process 40 at the specified destination internet address. A connection table (not shown) within the local compression process maintains the status of TCP/IP connections made or being attempted between local and remote internet address/port pairs. The local compression process 24 attempts to connect to the remote compression process 40 internet address specified in the local application's connection request. If the local compression process 24 does not have sufficient resources or resources available, such as free memory, an unused connection and/or does not support the selected algorithm, the attempt is aborted and the connection table modified accordingly. If a connection attempt fails, perhaps because the remote host is down or otherwise not available for connection, the local compression process maintains a decision table (like the decision table which the packet switch driver uses in determining if a given application's data should be diverted for compression), which is updated to indicate that connection packets destined for that internet address should not be diverted for compression. A timeout scheme and retry mechanism may be implemented so that the remote internet address may be tried at a later time, or the compression process may choose to route uncompressed data over a normal connection.

A second phase in communication of compressed data is the actual data transfer between the local and remote compression processes can proceed if the TCP/IP connection is effected. The local compression process 24 will send a buffer of data to the remote compression process 40 through the host TCP/IP. The data will contain application to application connection and addressing information, information relating to services to be provided by the local and remote compression processes and information concerning the selected compression algorithm and its use. The remote compression process receiving the data examines the various fields of information to determine if there is resident support for the facilities requested. If the remote compression process can not support the data, the connection will be aborted and the data will be returned to the local compression process with a response header indicative of the remote process facilities and the status of the connection.

The remote compression process, determining that there is support for the data transmission, tries to establish a connection to the remote application indicated by the destination address in the header. The remote compression process establishes the connection to the destination application, through the remote host packet switch driver 44, and the remote host TCP/IP 42. This is done in such a way that the address/port may be modified to present the appearance that the local application 10 was connected directly to the destination application in the remote Host B, without the intervening compression processes. This avoids the invalidation of the transmission by applications which selectively verify incoming port numbers for validity.

When the connection to the destination application is established, the remote compression process returns a response data buffer to the local compression process 40, which will indicate that the selected compression algorithm was accepted, and the status of the connection. When the local compression process 24 gets the response data buffer back from the remote compression process 40, it effects updating of the compression process connection and decision tables and the packet switch driver connection and decision tables. The information regarding address and selected compression algorithm are stored, preferably in the connection table, for use on the affected connection. The connection is completed locally, in the remote Host B. Host A starts compressing and transmitting data, while in Host B, the remote compression process decompresses transmitted data into the original data stream, performs TCP/IP processing in its internal TCP/IP to rebuild the data into packets and writes the data to the packet switch driver 44. The packet switch driver 44 injects the data, decompressed in accordance with the selected compression/decompression algorithm, back into the communication stream of the destination application.

If the connection between the remote compression process and the destination application fails, a response buffer is returned that is indicative f the error. When the local compression process gets the data buffer back with the indicated failure, it will effect closure of the TCP/IP connection and update the connection and decision tables accordingly, so that the connection proceeds normally.

The management utility 26 facilitates management and control of the configuration of compression paths. A system manager can define a configuration file 42, which is accessible to the management module 38 of a compression process to specify which data is selected for compression. Data is selectable by specifying particular network interfaces from which the packet switch driver will intercept packets for diverting to the compression process for compression. Compression paths can be specified, in a TCP/IP application of the invention, in accordance with the TCP/IP addressing scheme consisting of a destination internet address, destination port, source internet address and source port. Uni-directional or bi-directional compression can be specified. The configuration file is read upon initialization of the hereinbefore defined compression implementation.

Referring now to FIG. 7, the compression implementation according to the invention can be effected as a gateway implementation in a local area network having a plurality of processors, or gateway clients, that do not have such a compression implementation installed thereon. One host 50 having compression according to the invention implemented thereon acts as a gateway for a plurality of gateway clients 52 on the network to facilitate communication with a remote host 54 having such a compression facility. The gateway clients 52 must be known to the compression host 50, such as by establishing an indicative field in the configuration file, so that the gateway clients 52 can be serviced while other systems 56 on the network are precluded from similar services as described hereinbefore.

The compression/decompression module 34 modularly integrated in the compression process 24 (FIG. 6) can be replaced or supplemented with additional modular facilities to provide other services in the communication stream between layers of a given protocol stack. Services such as encryption/decryption could be made available to networked and internetworked computing machinery, according to the invention.

Although the invention as disclosed hereinbefore describes data compression in the context of networks according to TCP/IP, it can be appreciated that data compression can be implemented according to the invention between analogous layers in protocol suites other than TCP/IP, such as DECNET and OSI standard protocol suites and the like.

While data compression according to the invention is described in the context of compression for transmission on a wide area network, it will be appreciated that compression according to the invention is applicable in the context of local area networks. Especially with the trend in increased processor speeds which may result in compression processing speeds in excess of LAN transmission speeds.

Although selection of data for compression is effected hereinbefore in accordance with an address in the header, other data fields can be established and other criteria used for selection of data for provision of services as described.

One of ordinary skill in the art will appreciate that the invention described hereinbefore can be implemented in software, hardware or a combination thereof.

Although the invention has been shown and described with respect to an exemplary embodiment thereof, various other changes, omissions and additions in the form and detail thereof may be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. An apparatus for integrating services between protocol layers in a computer system having layered network protocols, comprising:

a first network layer having a first interface, and a second interface and operating according to a first protocol, and a second network layer having a first interface, and a second interface and operating according to a second protocol;

a switch driver responsive to said second interface of said first layer that receives a plurality of signal groups therefrom, said switch driver being configurable for routing at least some of said plurality of signal groups through at least one of a first path and a second path to said first interface of said second layer; and

at least one service module responsive to said switch driver for receiving at least some of said plurality of signal groups through said first path and that performs an operation on said at least some of said plurality of signal groups from said first path to form a plurality of modified signal groups,

wherein said at least one service module transmits said plurality of modified signal groups to said first interface of said first layer and said switch driver receives said plurality of modified signal groups.

2. The apparatus of claim **1** wherein said switch driver is selectively configurable in accordance with at least one set of predetermined criteria.

3. The apparatus of claim **2** wherein said at least one set of predetermined criteria is tested by comparing selected signals with a portion of signals of each of said plurality of signal groups.

4. The apparatus of claim **1** wherein said at least one service module comprises a compression processor and said operation performed on said at least some of said plurality of signal groups is data compression.

5. The apparatus of claim **1** wherein said at least one service module comprises a decompression processor and said operation performed on said at least some of said plurality of signal groups is data decompression.

6. The apparatus of claim **1** wherein said at least one service module operates according to a protocol compatible with said first protocol of said first layer.

7. The apparatus of claim **1** wherein said switch driver receives said plurality of modified signal groups for routing to said first interface of said second layer.

8. The apparatus of claim **1** wherein said at least one service module comprises an encryption processor and said operation performed on said at least some of said plurality of signal groups is data encryption.

9. The apparatus of claim **1** wherein said at least one service module comprises a decryption processor and said operation performed on said at least some of said plurality of signal groups is data decryption.

10. The apparatus of claim **1** wherein said protocol of said first layer requires disassembly of a data stream and assembly of said at least some of said plurality of signal groups therefrom and said at least one service module receives said at least some of said plurality of signal groups and reassembles said data stream.

11. The apparatus of claim **1** wherein said switch driver includes at least one set of predetermined criteria for determining said at least some of said plurality of signal groups for routing to said at least one service module.

12. The apparatus of claim **1** wherein said at least one service module includes at least one set of predetermined criteria for determining said at least some of said plurality of signal groups for routing to said at least one service module.

13. The apparatus of claim **1** wherein selected signals are stored in an accessible and modifiable configuration file.

14. The apparatus of claim **1** further comprising a path to pass a second plurality of signal groups from said first interface of said second layer to said second interface of said first layer and from said first interface of said first layer to said service module.

15. A method for providing a service between protocol layers in a computer system having layered network protocols, said method comprising the steps of:

intercepting a first plurality of signal groups travelling along a layered network protocol stack at a first layer interface;

comparing a portion of each of said first plurality of signal groups with a predetermined set of signal groups to determine selected signal groups for performing an operation thereon to provide said service;

routing said selected signal groups to a service module;

performing said operation on said selected signal groups to provide said service; and

routing said selected signal groups to a second layer interface.

16. The method of claim **15** wherein said operation is data compression.

17. The method of claim **15** wherein said operation is data decompression.

18. The method of claim **15** wherein said operation is data encryption.

19. The method of claim **15** wherein said operation is data decryption.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	:	5,307,413
DATED	:	April 26, 1994
INVENTOR(S)	:	Philip C. Denzer

It is certified that error appears in the above-indentified patent and that said Letters Patent is hereby corrected as shown below:

Column 3, line 5, "streams after" should read --streams <u>after</u>--.

Column 7, line 16, "processes can" should read --processes, which can--.

Signed and Sealed this

Third Day of January, 1995

Attest:

[illegible]

BRUCE LEHMAN

Attesting Officer *Commissioner of Patents and Trademarks*